UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

AMBIENTE INTEGRADO PARA SÍNTESE DE CONTROLADORES NEURAIS ADAPTATIVOS

Dissertação submetida à Universidade Federal de Santa Catarina para a obtenção do grau de **Mestre em Engenharia Elétrica**

MAURO CEZAR KLINGUELFUS

Florianópolis, 30 de Abril de 1996

Ambiente Integrado para Síntese de Controladores Neurais Adaptativos

Mauro Cezar Klinguelfus

Esta dissertação foi julgada adequada para obtenção do título de
Mestre em Engenharia na especialidade **Engenharia Elétrica**,
área de concentração **Sistemas de Controle, Automação e Informática Industrial**,
e aprovada em sua forma final pelo Curso de Pós-Graduação.

Prof. Dr.-Ing. Marcelo Ricardo Stemmer
Orientador

Prof. Dr. Ênio Valmor Kassick
Coordenador do Curso de Pós-Graduação em
Engenharia Elétrica

Banca Examinadora

Prof. Dr.-Ing. Marcelo Ricardo Stemmer

Prof. Dr. Edson Roberto de Pieri

Prof. Dr. Antônio Augusto Rodrigues Coelho

Prof. Dr. Jorge Muniz Barreto

Agradecimentos

Ao amigo e companheiro Marcelo Ricardo Stemmer, pela competência e interesse com que orientou este trabalho.

Ao amigo Daniel Pagano cuja a brilhante orientação e incentivo permitiu que fosse possível levar este trabalho a contento.

Ao amigo Heitor Silvério Lopes pela amizade e colaborações técnicas, contribuiram sensivelmente para a realização deste trabalho.

Aos professores do curso de pós-graduação do LCMI, pelo incentivo, compreensão e apoio.

Aos colegas de pós-graduação pelo companherismo e amizade.

Ao amigo William Lopes de Oliveira, pela particular colaboração na solução de problemas de programação.

A COPEL e a CAPES que criaram as condições de suporte financeiro para o desenvolvimento deste trabalho.

A meus pais, que com ternura e afeição, incentivaram sempre os meus estudos.

A minha esposa Salete Maria Bettin Klinguelfus e a meus filhos Gustavo e Taline Klinguelfus, que sempre me apoiaram e estimularam a minha carreira profissional com apreço e carinho.

Índice

1. Introdução Geral 1

2. Redes Neurais Artificiais 5

2.1 Conceitos Fundamentais 5

2.2 Conceituação Biológica 6

2.3 Modelagem do Neurônio 8

3. Redes Multicamadas 13

3.1 Introdução 13

3.2 Algoritmo de Treinamento "Backpropagation" 15

3.3 Algoritmos Genéticos 22

3.3.1 Introdução 22

3.3.2 Descrição do Algoritmo 23

3.4 Coexistência dos Algoritmos 31

3.5 Aspectos Gerais do Treinamento 35

3.5.1 O Problema dos Mínimos Locais 35

3.5.2 Validação 38

3.5.3 Sobretreinamento 38

3.5.4 Entendendo o Papel do Neurônio 40

4. Redes Neurais Aplicadas à Área de Controle 43

5. Projeto e Implementação de Redes Neurais Multicamadas 48

5.1 Projetando-se uma Rede Neural Multicamadas 48

5.2 Interpretando os Pesos 53

5.3 Projetando-se o Conjunto de Treinamento 55

5.4 Implementação 58
5.4.1 Algoritmos tipo Gradiente Descendente 58
5.4.2 Algoritmo Genético 61

5.5 Pré-treinamento 64
5.5.1 Introdução 64
5.5.2 Desenvolvimento 64

5.6 Aquisição On-Line 66

5.7 Realização do Controlador 70
5.7.1 Introdução 70
5.7.2 Controle Indireto 70
5.7.3 Controle Direto 74

6. Testes do Ambiente no Controle de uma Planta Piloto 77

6.1 Descrição da Planta Piloto 77

6.2 Testes Realizados e Resultados Obtidos 79

6.3 Conclusões dos Ensaios 82

7. Conclusão 84

8. BIBLIOGRAFIA 88

ANEXO A - CARACTERÍSTICAS GERAIS DO AMBIENTE 96

ANEXO B - UTILIZAÇÃO DO AMBIENTE 98

B.1 MODO L(earning) 99

B.2 MODO O(utput generation) 103

B.3 MODO D(irect control) 103

B.4 MODO I(ndirect control) 109
B.4.1 Introdução 109
B.4.2 Rede identificadora 110
B.4.3 Rede controladora 117

B.5 Definição dos Limites das Entradas e Saídas 119

B.6 "SETUP" (Controle Direto e Indireto) 119

B.7 Menu de Opções 122

B.8 Treinamento 125
B.8.1 Treinamento Genético 126
B.8.2 Treinamento "Backpropagation" 128
B.8.3 Apresentação do Resultado do Treinamento. 130

B.9 Obtenção dos "TRUE VECTORS" 131

B.10 Captura Dinâmica 133

B.11 Compilação, Ambiente de Programação e Hardware 134

Lista das Figuras

FIGURA 1 - NEURÔNIO BIOLÓGICO 7
FIGURA 2 - NEURÔNIO ARTIFICIAL 9
FIGURA 3 - FUNÇÕES DE ATIVAÇÃO 11
FIGURA 4 - REDE MULTICAMADAS 14
FIGURA 5 - SUPERFÍCIE DA FUNÇÃO DO ERRO 36
FIGURA 6 - TREINAMENTO & TESTE 39
FIGURA 7 - PAPEL DE CADA NEURÔNIO 42
FIGURA 8 - DERIVADOR NUMÉRICO 44
FIGURA 9 - ERRO TREINAMENTO & TESTE 51
FIGURA 10 - SEQÜÊNCIA DE TREINAMENTO 53
FIGURA 11 - CONFIGURAÇÃO PARA AQUISIÇÃO DOS "TRUE VECTORS" 65
FIGURA 12 - TABELA DE "TRUE VECTORS" 65
FIGURA 13 - ACRÉSCIMO OFF-LINE DE VETORES DE TREINAMENTO 65
FIGURA 14 - AQUISIÇÃO ON-LINE DE NOVOS VETORES DE TREINAMENTO 66
FIGURA 15 - TABELA AMPLIADA COM VETORES ADQUIRIDOS ON-LINE 67
FIGURA 16 - AQUISIÇÃO DE VETORES DE TREINAMENTO PARA A REDE NEURAL 71
FIGURA 17 - TREINAMENTO DA RNI 71
FIGURA 18 - ESQUEMA PARA CONTROLE INDIRETO 72
FIGURA 19 - ESQUEMA PARA CONTROLE DIRETO 75
FIGURA 20 - PLANTA PILOTO: GERADOR 78
FIGURA 21 - RELAÇÃO SAÍDA/ENTRADA DA PLANTA 78
FIGURA 22 - PID & CONTROLADOR NEURAL 79
FIGURA 23 - EFEITO DA VARIAÇÃO DA VELOCIDADE 80
FIGURA 24 - EFEITO "DELAY ACCELERATOR" 81
FIGURA 25 - VARIAÇÃO DA VELOCIDADE PÓS-TREINAMENTO 82
FIGURA 26 - DESEMPENHO DA REDE IDENTIFICADORA EM RELAÇÃO A PLANTA 82
FIGURA 27 - TELA INICIAL 99
FIGURA 28 - TREINAMENTO OFF-LINE 100
FIGURA 29 - CONTROLE DIRETO 104
FIGURA 30 - ATUAÇÃO DO "BASE FACTOR" 106
FIGURA 31 - CONTROLE INDIRETO 110
FIGURA 32 - REDE IDENTIFICADORA 111
FIGURA 33 - "SETUP" DA REDE IDENTIFICADORA 111
FIGURA 34 - OBTENÇÃO DOS VETORES DA REDE IDENTIFICADORA 112
FIGURA 35 - CAPTURA DE VETORES DA REDE IDENTIFICADORA 114
FIGURA 36 - TREINAMENTO E TESTE DA REDE IDENTIFICADORA 115
FIGURA 37 - TREINAMENTO DA REDE IDENTIFICADORA 115
FIGURA 38 - TESTANDO A REDE IDENTIFICADORA 116
FIGURA 39 - REDE CONTROLADORA (CONTROLE INDIRETO) 118
FIGURA 40 - TELA "SETUP" PARA CAPTURA DINÂMICA 120
FIGURA 41 - MENU TEMPO REAL 124
FIGURA 42 - TREINAMENTO GENÉTICO 127
FIGURA 43 - TREINAMENTO "BACKPROPAGATION" 129
FIGURA 44 - OBTENÇÃO DOS "TRUE VECTORS" 131

Resumo

Este trabalho descreve um Ambiente baseado em redes neurais com treinamento neuro-genético para realização de controladores neurais adaptativos independendo da complexidade do processo a controlar e não requerendo um modelo matemático para o projeto. Possue todas as ferramentas necessárias para a aquisição dos vetores de treinamento, a execução do treinamento propriamente dito e realização do controlador em tempo real para diferentes taxas de amostragem. É apresentado um resumo da teoria que embasou o desenvolvimento bem como os procedimentos para utilização do Ambiente implementado.

Resultados práticos são também apresentados para uma planta piloto. Análises comparativas são realizadas com a utilização da solução apresentada frente a outras que também utilizam técnicas de redes neurais e PIDs discretos.

O bom desempenho dos resultados práticos obtidos demostram a potencialidade do Ambiente e dos métodos adotados.

Abstract

This work describes a neural network based environment with neuro-genetic training for adaptive neural control, for any process complexity, and doesn't require knowledge about the mathematical model of the process. It includes all the necessary resources for the training acquisition vectors, for the neural network training and for real time control of the plant with different sampling rates. It presents also a summary of the theory on which the development was based as well as the procedure adopted in the use of the environment.

The work also includes test results obtained with the environment when controlling a power generator. The capabilities of this solution are shown and compared to other well known techniques, including PIDs.

The results obtained expose the good performance of the environment and adopted procedures.

1. Introdução Geral

O uso de Redes Neurais Artificiais (RNA) em sistema de controle se mostra cada vez mais como uma alternativa interessante, principalmente quando defronta-se com plantas razoavelmente complexas e difíceis de serem modeladas. Nestas condições, os clássicos controladores baseados em PID apresentam como principal dificuldade justamente a modelagem matemática do problema que nem sempre se apresenta de uma maneira trivial.

Dentre alguns aspectos que motivam a utilização de redes neurais, pode-se citar os seguintes:

- Método não-algorítmico: É um método intuitivo, ou seja, para o usuário é mais importante o conhecimento que este tem em relação ao seu próprio problema do que um conhecimento prévio sobre as técnicas voltadas a utilização de redes neurais.
- A rede neural aprende através de exemplos, o que permite que ela represente um determinado comportamento. Nestes exemplos são apresentados para a rede a relação entrada/saída que esta deve representar.
- Simplicidade: Ao contrário do que pode-se imaginar, a estrutura interna de uma rede neural artificial é algo muitas vezes mais simples do que o observado nos seres vivos.
- Generalização: Uma rede neural é capaz de responder a padrões para os quais não necessariamente foi treinada. Basta que durante o treinamento tenha-se a preocupação de escolher exemplos para o treinamento que atendam a faixa de interesse.
- Resposta rápida: Uma vez treinada, a rede apresenta um tempo de resposta rápido, tornando-a interessante para aplicações em tempo real.
- Aproximador universal: Com uma rede neural é possível representar qualquer função matemática.
- Eliminação natural de ruídos: Isto ocorre devido a própria característica construtiva da rede.
- Conhecimento mínimo do processo: Controladores de processos usando redes neurais necessitam de um mínimo conhecimento do modelo matemático que

envolve o processo. Também ocorre que as características intrínsecas do processo serem automaticamente consideradas.

Neste aspecto, redes neurais artificiais facilitam a realização de controladores, visto não estarem calcadas sobre os métodos clássicos de elaboração de controladores, e sim, estarem baseadas num processo de aprendizado que deve ser iniciado numa fase preliminar à realização do controlador propriamente dito. O que se deseja dizer com isto é que, somente necessita-se saber de antemão as relações de entrada/saída do processo a controlar, ou seja, deve-se ter o prévio conhecimento de alguns conjuntos de valores de grandezas que definam um conjunto de entradas e suas respectivas saídas (treinamento supervisionado). Cada conjunto entrada/saída corresponde a um vetor de treinamento e o conjunto de vetores de treinamento correspondem a um padrão de treinamento. A quantidade de vetores de teste de cada padrão é proporcional, de uma certa forma, à complexidade do processo a se controlar. Durante os ensaios, observou-se que uma quantidade razoável de vetores de treinamento para a maioria dos problemas práticos é em torno de 20 a 60.

A inspiração para criação das denominadas redes neurais artificiais remonta aos anos 40 e é baseada nos modelos biológicos. Entretanto, somente nas últimas décadas é que o interesse por estes sistemas conexionistas cresceu de uma maneira sólida graças à melhor compreensão de determinados problemas e limitações que existiam, além do desenvolvimento tecnológico dos sistemas computacionais. O surgimento de novas estruturas e algoritmos de aprendizado e a disponibilidade de processadores cada vez mais rápidos, conseguiram de uma certa forma, popularizar o uso destas redes para as mais diferentes aplicações, destacando-se além das aplicações em controle, um quase interminável conjunto de outras utilizações como processamento de alarmes, reconhecimento de padrões como escrita, imagem e voz, em predição de eventos, em deteção de faltas e diagnóstico, processamento digital de sinal, despacho econômico de carga, etc. [2,3,10,11,12,40,42,43,44,46].

Um estimador sem modelo. É assim que podem ser vistas as redes neurais, por serem aproximadores universais de funções gerais, que permitem mapear vetores de entrada em vetores de saída sem necessitar de um modelo matemático para tal.

Este trabalho se propõe a apresentar uma alternativa viável e prática para a realização de controladores de processos em tempo real que possuam ou não linearidades na sua faixa de atuação, e principalmente, quando a modelagem matemática em questão não é trivial ou mesmo praticamente impossível de ser obtida. O trabalho como um todo culmina em um ambiente de hardware e software implementado para plataformas IBM-PC ou compatíveis, integrando todas as etapas necessárias para o desenvolvimento completo do controlador incluindo facilidades para realizações de testes específicos, e ainda, mecanismos de segurança de controle para o caso que o ambiente desenvolvido venha a ser utilizado como o próprio controlador. Em outras palavras, é possível usar o ambiente desenvolvido como o controlador do processo em tempo real, ou então reproduzir somente as redes utilizadas de maneira independente do restante do ambiente.

Dentre as soluções que foram adotadas são apresentadas algumas que ainda não foram observadas em outras propostas de soluções de problemas semelhantes. Gostaríamos de destacar o método de controle direto que está sendo originalmente apresentado neste trabalho.

Ainda utiliza-se o termo redes neurais ou simplesmente redes em substituição à redes neuronais artificiais. Também é indicado entre aspas duplas todas as palavras em inglês que não tiverem sido ainda incluídas no vocabulário utilizado no Brasil. As menções ao Ambiente Integrado para Síntese de Controladores Neurais Adaptativos são feitas somente usando-se a palavra Ambiente, mantendo-se a primeira letra maiúscula.

No capítulo 2 é descrito o neurônio como a unidade elementar de uma rede neural, destacando tanto o neurônio biológico como o artificial.

O capítulo 3 se refere às características implícitas de uma rede neural artificial enfocando e justificando a utilização das redes multicamadas. São descritos os algoritmos de treinamento utilizados e os aspectos gerais do treinamento destas redes.

A discussão da aplicação de redes neurais em controle é feita de maneira mais detalhada no capítulo 4. Neste capítulo são mencionadas técnicas distintas de

redes neurais existentes na literatura justificando-se as razões que levaram a adoção da rede multicamadas frente as essas outras técnicas.

Todos os aspectos construtivos de uma rede neural aplicada à controle são enfatizados no capítulo 5. Também são destacados os modelos de controle direto e indireto adotados neste trabalho.

No capítulo 6 são descritos a planta piloto utilizada para testar o Ambiente, como também os ensaios realizados.

Em seguida são apresentadas as sugestões para continuidade do trabalho e as conclusões gerais.

Finalmente, apresenta-se os anexos que têm como objetivo a descrição completa do Ambiente no que tange à sua utilização como uma ferramenta prática de trabalho.

O anexo A contém um resumo das características gerais permitindo uma rápida visualização do potencial de utilização do Ambiente.

No anexo B são apresentadas de uma maneira detalhada todas as funções, comandos e opções disponíveis nas várias telas que foram implementadas no Ambiente como um todo. Basicamente este anexo pode ser descrito como o manual de utilização do Ambiente.

2. Redes Neurais Artificiais

2.1 Conceitos Fundamentais

Redes neurais artificiais são uma nova classe de sistemas computacionais formado por dezenas, centenas e até mesmo milhares de elementos simples como se fossem pequenas células, onde cada elemento procura simular o comportamento do neurônio biológico. Estes elementos são interconectados entre si de maneira similar ao que ocorre no cérebro humano. A forma como os neurônios são interligados confere as características de aprendizado da rede neural artificial, como também ocorre nos seres vivos. Seguindo este raciocínio, para cada aplicação deve-se buscar estruturas que mais se adaptem a necessidade de um determinado problema.

Na literatura [2,3,42] encontra-se vários tipos distintos de estruturas ou topologias onde se ressaltam as potencialidades de cada uma delas. Na realidade o que se busca no estudo das redes neurais artificiais é reproduzir comportamentos similares aos encontrados na própria natureza, onde se considera que os processos naturais são suficientemente otimizados a ponto de servirem de referência para os desenvolvimentos de novos algoritmos aplicáveis às redes neurais artificiais.

Atualmente existe uma forte tendência de se rejeitar estruturas que apesar de apresentarem comportamento externo semelhante aos comportamentos naturais mas que desrespeitam este paradigma no que tange ao processamento interno. Um exemplo, é a utilização de neurônios não-monotônicos, o que é abordado mais detalhadamente ao longo deste trabalho.

Numa grande parte das soluções adotadas, pode-se sempre observar o forte apelo resultante de comportamento já conhecido na natureza, ou mesmo, daqueles comportamentos que resultam como produto de alguma intervenção

humana, mas que são respeitadas as características físicas dos processos envolvidos.

Máquinas que pensam. Até pouco tempo eram simplesmente produto da ficção. Talvez o termo pensar seja um pouco exagerado. O que se espera realizar com as redes neurais artificiais são máquinas que aprendam um comportamento, e como extensão disto, que sejam capazes de se adaptarem ao meio a que estão acopladas.

Um aspecto curioso das redes neurais artificiais é que, de maneira similar ao que ocorre com os seres vivos, a capacidade de aprendizado não necessariamente define um comportamento determinístico, conferindo um certo grau de não-previsibilidade. Desta forma, o comportamento resultante será tão próximo ao esperado quanto melhor for a qualidade do processo de treinamento utilizado.

Quase sempre treinar significa apresentar à rede um conjunto de entradas e o respectivo comportamento de saída. Talvez este seja o maior desafio que enfrenta-se quando trabalha-se com redes neurais artificiais.

2.2 Conceituação Biológica

Embora ainda se conheça pouco sobre a operação completa do cérebro humano, as redes neurais são baseadas nesta estrutura.

O sistema nervoso humano é constituído de células denominadas de neurônios. Estima-se 10^{11} neurônios participando em aproximadamente 10^{13} interconecções sobre vias de transmissão que podem variar de um ou mais metros. Cada neurônio compartilha muitas características com outras células do corpo, mas uma capacidade única para receber, processar e transmitir sinais eletroquímicos sobre as vias neurais do sistema de comunicação do cérebro.

Para compreender as características básicas dos neurônios biológicos, é necessário identificar as estruturas e mecanismos fundamentais presentes no sistema nervoso, tais como: corpo da célula (ou soma), dendrito, sinapse, axônio, etc., como mostra a figura 1. A partir da observação de variações nas

estruturas e comportamento dos neurônios surge um conjunto de tipos de neurônios em vez de um único tipo.

Dendritos prolongam-se do corpo da célula para outros neurônios onde eles recebem sinais num ponto de conexão denominado de sinapse. No lado da recepção da sinapse, estas entradas são conduzidas para o corpo da célula, onde algumas entradas tendem a excitar a célula e outras tendem a inibir seu disparo. Do soma, a partir de um ponto denominado de hillock, surge um filamento comprido denominado de axônio. Na extremidade do axônio tem-se algo como micro-ramificações denominadas de arborização axonial. Os pontos extremos da arborização axonial são denominados de terminais do neurônio ou botões. Quando a excitação acumulada na célula excede a um limiar ("threshold"), a célula dispara, enviando um sinal através do axônio para outros neurônios conectados neste terminais.

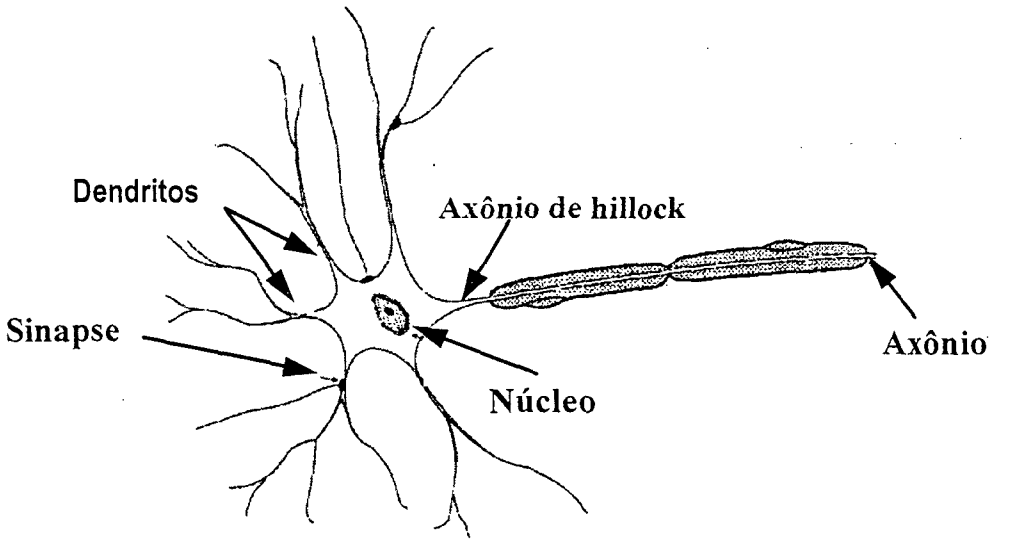


Figura 1 - Neurônio biológico

O esquema de conexão não se resume necessariamente à conexão dos terminais de um neurônio aos dendritos de outros neurônios. É possível que os terminais dos neurônios estejam conectados a outros terminais do próprio ou de outros neurônios, aos dendritos do próprio neurônio, ou diretamente ao corpo da célula do próprio ou de outros neurônios, ou ainda de dendrito para dendrito. Alguns neurônios apresentam um padrão de conexão bastante seletivo e específico, onde os terminais dos neurônios devem ser conectados a pontos específicos,

como por exemplo, os corpos das células dos neurônios que controlam a ação motora dos dedos do pé que estão situados na espinha dorsal e seu axônio percorre toda a perna até atingir os dedos.

O axônio pode transmitir eletricidade em ambas as direções, mas o fluxo de informação sempre vai no sentido dos dendritos para o axônio, isto porque o impulso elétrico começa no lado do hillock. Em condições normais, se uma corrente elétrica é aplicada em um ponto do axônio, o seu potencial decai exponencialmente a medida que a corrente é propagada através do axônio.

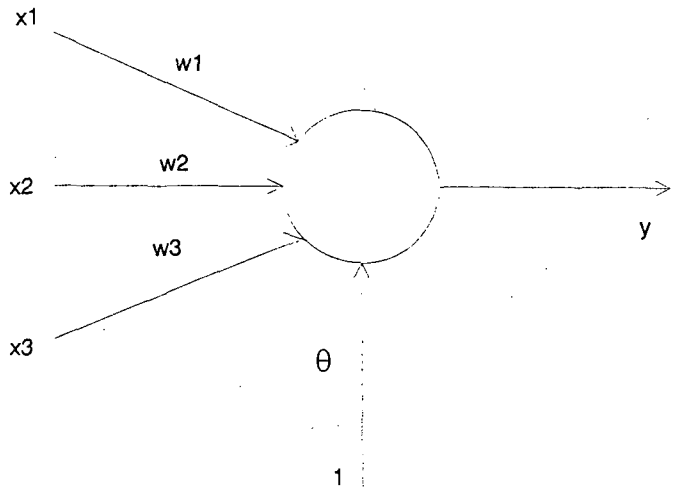
O mecanismo de disparo define uma característica *tudo ou nada* dos neurônios, uma vez que em qualquer situação o neurônio pode estar disparado ou não. O processo de excitação de um neurônio corresponde à integração de várias entradas, sendo que cada uma delas em particular pode estar contribuindo para excitar o neurônio ou inibí-lo.

Existe ainda um intervalo de tempo entre dois impulsos, conhecido como período refratório, no qual o impulso não pode ser gerado e propagado até que ocorra a restauração química dentro do neurônio.

2.3 Modelagem do Neurônio

De uma forma genérica, pode-se dizer que uma rede neural se constitui de modelos de processamento distribuído e paralelo. A unidade básica de uma rede neural é o neurônio (figura 2). Estes neurônios têm a capacidade de mudar o seu comportamento após um treinamento dinâmico. Estes elementos são interconectados por ligações com valores variáveis também denominados de pesos.

Basicamente um neurônio corresponde a uma soma ponderada de entradas, soma esta aplicada a uma função de ativação.



$$net = \Sigma_i x_i w_i + \theta$$
$$y = f(net)$$

Figura 2 - Neurônio Artificial

Observando a figura 2, verifica-se que a soma dos valores das entradas *Xi* multiplicadas pelos seus respectivos pesos *wi*, adicionada a um valor limite θ conhecido como "threshold", é por sua vez aplicada a função de ativação. Em geral a função de ativação é uma função não-linear, contínua e diferenciável.

Na literatura especializada encontra-se designações de vários modelos de neurônios além do modelo biológico apresentado neste trabalho, outros como o neurônio RAM, neurônio PLN, neurônio GSN, cada um com sua especificidade [2,3,42].

Neurônio de McCulloch e Pitts

McCulloch e Pitts [4] propuseram um modelo simplificado de neurônio biológico. O modelo baseia-se no fato de que, em um dado instante de tempo, o neurônio ou está disparado ou está inativo imprimindo desta forma, um comportamento discreto e binário. Utilizando-se, então, o cálculo proposicional temporal, pode-se modelar o comportamento de neurônios biológicos. O neurônio foi definido como tendo dois estados, disparado ou inativo, que corresponde ao verdadeiro e falso do cálculo da lógica proposicional de ordem zero, ou ainda, o zero e um da álgebra booleana.

Neste neurônio existem conexões excitatórias e inibitórias representadas através de um *peso* com sinal, o qual reforça ou dificulta a geração de um impulso de saída. Um neurônio *Ni* produz um impulso, ou seja, uma saída *oi* = 1, se e somente se, a soma das entradas é maior ou igual a um limiar. A equação 1 define a função de saída do neurônio de McCulloch-Pitts:

$$O_i(x)\begin{cases}1 & \text{if } x = \sum_j w_{ij} i_{ij} - \theta i \geq 0 \\ 0 & \text{if } x = \sum_j w_{ij} i_{ij} - \theta i < 0\end{cases} \qquad (1)$$

Onde w_{ij} é o peso da conexão associado com a entrada i_{ij}, e θ_i é o limiar de ativação do neurônio n_i.

A partir do modelo proposto por McCulloch e Pitts foram derivados vários outros modelos que permitem a produção de uma saída qualquer, não necessariamente zero ou um. Também surgiram várias definições quanto a forma de ativação ou função de ativação como mostra a figura 3, onde são ilustradas quatro funções de ativação, entre elas: a função linear, a função rampa, a função degrau e a função sigmoidal.

A função sigmoidal também conhecida como "S-shape" ilustrada na figura 3.d, é uma função semilinear, limitada e monotônica. É possível definir várias funções sigmoidais. Uma das funções sigmoidais mais importante é a função logística definida pela equação 2.

$$S_i(x) = \frac{1}{1 + e^{-x/T}} \qquad (2)$$

O parâmetro T determina o formato da curva.

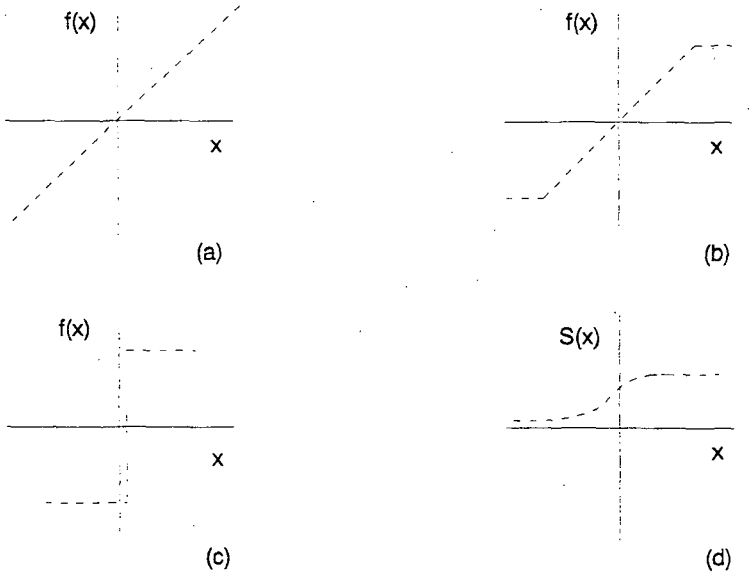


Figura 3 - Funções de ativação

O motivo que tornou esta a função de ativação mais utilizada, é a facilidade de se obter a derivada, que é dada pela seguinte equação:

$$S_i'(x) = S_i(x)(1 - S_i(x)) \tag{3}$$

Outras funções sigmoidais, como a tangente hiperbólica e o arcotangente são algumas vezes utilizadas. A equação seguinte apresenta a função tangente hiperbólica.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{4}$$

Na maioria dos casos, a forma exata da função faz pouco efeito durante a utilização propriamente dita da rede, entretanto pode ter um impacto maior na velocidade de treinamento.

Em [3] reporta-se que valores relativamente pequenos de derivada de uma função de ativação logística tornam o treinamento lento para o algoritmo de "backpropagation", como descreve-se nos capítulos seguintes. Nas equações 5 e 6 são apresentadas duas alternativas para atenuar este problema.

Neste trabalho foram ensaiadas as equações (5) a título de comparação de desempenho. Observou-se que o desempenho foi muito pouco afetado. Sendo assim, optou-se por manter a função de ativação logística e fazer o controle dos valores pequenos de derivada através do método proposto no capítulo 3, principalmente em razão do esforço computacional. Pelo mesmo motivo, ou seja, tempo de processamento, não foram feitos testes com as equações (6), sendo somente apresentadas aqui a título de informação.

$$f(x) = \frac{2}{\pi}\tan^{-1}(sinh(x))$$
$$f'(x) = \frac{2}{\pi}\sec h(x) \qquad (5)$$

$$f(x) = \frac{2}{\pi}\left(\frac{\tanh(x)}{\cosh(x)} + \tan^{-1}(sinh(x))\right)$$
$$f'(x) = \frac{4}{\pi}\sec h^3(x) \qquad (6)$$

É importante observar que a função sigmóide nunca atinge o seu valor máximo e mínimo teórico. Um neurônio com uma função logística é considerado totalmente ativado quando a saída está em torno de 0,9 e desativado quando está em torno de 0,1. Desta forma não é razoável esperar saídas próximas ou iguais a 0 ou a 1.

A rede neural é definida pela estrutura formada pelas interconexões entre neurônios, pelo método de treinamento adotado, entre outras coisas. O processo de treinamento define um conjunto de pesos iniciais e como estes pesos devem ser ajustados.

3. Redes Multicamadas

3.1 Introdução

Inicialmente, dentro do contexto de redes neurais, aborda-se neste trabalho os tópicos que são efetivamente utilizados no Ambiente proposto. Demais tópicos gerais sobre este tema podem ser pesquisados na literatura especializada [2,3,42,43].

Dentre as várias topologias de redes até hoje descritas, as redes multicamadas "feedfoward" apresentada inicialmente por Rumelhart, Hinton e Williams em 1986 [3,42], têm sido vistas como um efetivo sistema de aprendizado bastante indicado para aplicações na área de controle como também em outras áreas. Nesta rede, cada conjunto de nós é associado em camadas. A saída dos nós de uma camada são transmitidos aos nós das outras camadas através de uma conexão que amplifica, atenua ou inibe esta conexão por meio de elementos denominados pesos. Exceto na camada de entrada, a entrada de cada nó da rede recebe a soma das saídas dos nós da camada anterior, ponderada pelos respectivos pesos. Desta forma, cada nó é ativado de acordo com sua entrada e a função de ativação utilizada, como mostra a figura 4.

Nesta estrutura é acrescentado um neurônio a cada camada intermediária e a camada de saída. A entrada destes outros neurônios é mantida sempre em 1 (um) o que confere uma característica de estabilidade durante a fase de treinamento. O peso correspondente a esta entrada é denominado de *"threshold"*. Caso o *"threshold"* seja positivo, seu efeito é provocar um deslocamento da função de ativação para a esquerda ao longo do eixo horizontal (figura 3.d). Efeito interessante ocorre quando, na função de ativação divide-se o valor correspondente dos somatórios ponderados das entradas por um fator positivo qualquer. Quanto maior é este fator tanto mais suave é a inclinação resultante da função sigmoidal. Já, quando este fator é pequeno, a

resultante da função de ativação tende a assemelhar-se mais ao desenho de uma função degrau.

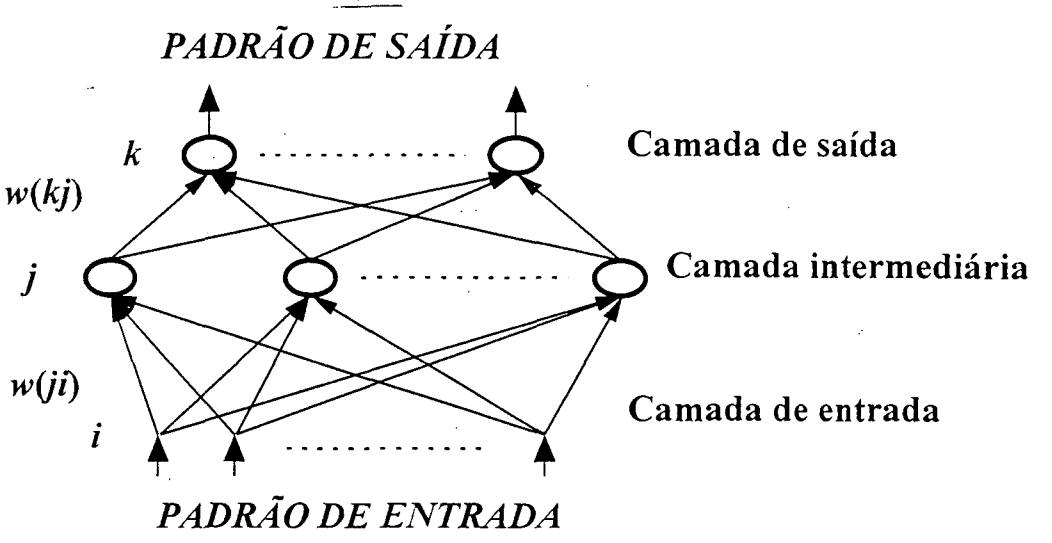


Figura 4 - Rede Multicamadas

Constituída a rede, ou seja, definida a sua completa estrutura, o passo seguinte é a identificação do método de treinamento mais adequado para a rede escolhida.

Um dos algoritmos que obteve mais sucesso no treinamento de redes multicamadas tem sido o "backpropagation" ou algoritmo de propagação reversa. Este algoritmo vem a ser uma generalização do algoritmo dos mínimos quadrados. Vários outros algoritmos de treinamento foram desenvolvidos no sentido de otimizar o desempenho deste algoritmo, no entanto, na maioria das vezes são versões modificadas ou incrementadas do "backpropagation" [3,5,15,19,24,25,33,41].

Este algoritmo tem como inconveniente a sua alta sensibilidade aos denominados mínimos locais, como é exaustivamente tratado ao longo deste trabalho. Como conseqüência, outros algoritmos foram desenvolvidos com o intuito de atenuar este problema. Entretanto, não existe até hoje um único algoritmo que seja totalmente imune a problemática dos mínimos locais. No capítulo 5 é tratado de um algoritmo denominado "simulated annealing" que vem a ser basicamente uma extensão do "backpropagation" que procura atenuar o efeito dos mínimos locais [24].

Na busca de formas de contorno ao problema destes mínimos locais, a técnica que mais vem obtendo sucesso é a denominada de algoritmos genéticos ou simplesmente AGs [3,26]. Esta técnica segue o conceito de reprodução genética e se caracteriza principalmente por apresentar um espaço de busca mais amplo, contudo, comparativamente a outros algoritmos, não é muito eficiente na busca do mínimo efetivo da função.

Como solução, a prática nos diz que é mais interessante fazer uma composição dos dois algoritmos, resguardando os AGs para a fase preliminar do treinamento e delegando ao "backpropagation" a missão de buscar o mínimo efetivo da função, mantendo desta forma uma solução híbrida.

Nos itens seguintes descreve-se com detalhes o funcionamento destes dois algoritmos.

3.2 Algoritmo de Treinamento "Backpropagation"

"Backpropagation", é um processo iterativo variante do método proposto por Rumelhart, Hinton e Williams em 1986, baseado numa regra conhecida como *regra delta generalizada* [44]. Foi o primeiro método prático para treinamento de rede "feedfoward" multicamadas. Este algoritmo é um processo iterativo e na sua forma básica, enquadra-se dentro dos algoritmos de gradiente descendente. Neste algoritmo é computado o gradiente da função erro, ou seja, o erro da rede para um determinado conjunto de treinamento. Desta maneira, são dados passos na direção do gradiente negativo e repetido tantas vezes quanto forem necessários. Desde que os passos dados sejam no sentido de reduzir o erro, o que se espera é atingir a localização do erro mínimo. A distância do passo, denominada de taxa de aprendizagem, pode ser crítica se não houver algum procedimento de ajuste automático. Se o passo é pequeno, a convergência ocorre lentamente, e se grande, pode nunca atingir o valor do erro esperado [2,3,5,10,15,16,42,43].

Uma consideração a ser feita é sobre a utilização de um algoritmo de gradiente descendente ("backpropagation") em vez do método gradiente conjugado que é considerado genericamente mais robusto e até rápido [3,42]. Este segundo método, apresenta entretanto, o mesmo problema do "backpropagation" quanto

aos denominados mínimos locais mostrados na figura 5. O "backpropagation", sem dúvida, apresenta-se como um algoritmo mais popular e de fácil implementação, além de ter sido mais explorado que os demais. Estas são algumas das razões da adoção do algoritmo "backpropagation".

Durante a fase de treinamento, é apresentado um padrão para a rede na forma de uma entrada e se busca fazer o ajuste das conexões ou dos pesos de forma a se obter um padrão de saída correspondente ao desejado. Este ajuste é repetido para outros padrões entrada/saída até que o ajuste final conseguido, ou seja, um único conjunto de pesos e *threshold* satisfaçam a todos os padrões apresentados à rede dentro de uma tolerância aceitável.

Em geral, as saídas $[o_{pk}]$ não são exatamente iguais aos valores desejados $[t_{pk}]$. Sendo assim, para cada padrão é calculado o erro quadrático médio dado pela expressão:

$$E_p = \frac{1}{n} \sum_k (t_{pk} - o_{pk})^2 \tag{7}$$

O erro do sistema ou para o conjunto de padrões é definido como:

$$E = \frac{1}{m} \sum_p E_p \tag{8}$$

onde k corresponde ao número de neurônios da última camada, p corresponde ao número de padrões, n número de neurônios da camada de saída e m número de vetores de treinamento ou padrões.

No processo de treinamento, procura-se achar uma maneira rápida de se reduzir o erro E_p. Diferentes resultados são obtidos quando se busca trabalhar sobre E ou E_p. De qualquer forma, as correções dos pesos são feitas de maneira seqüencial, sendo que o aprendizado é feito aplicando-se um padrão a cada instante. O que se busca, de forma geral, é a minimização da função do erro E.

A convergência é conseguida provocando-se uma variação incremental Δw_{kj} nos pesos e threshold, proporcional a $-\partial E/\partial w_{kj}$, tal que:

$$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} \tag{9}$$

O termo η também conhecido como taxa de aprendizagem, permite definir uma velocidade de convergência versus granularidade de aproximação, ou seja, quanto maior é este valor, tanto maior é a variação incremental sobre os pesos e "thresholds", em contrapartida, tem-se uma maior dificuldade de se aproximar do ponto de convergência ótimo devido à distância que se estabelece entre cada um dos passos de minimização da função do erro.

O erro E é expresso em termos das saídas o_k, ou seja, da saída não-linear de cada nó k, tal como:

$$o_k = f(\text{net}_k) \tag{10}$$

onde net_k é a entrada do k-ésimo nó, e por definição é a soma linear ponderada de todas as saídas dos neurônios da camada anterior:

$$\text{net}_k = \Sigma\, w_{kj}\, o_j \tag{11}$$

A derivada parcial $\partial E/\partial w_{kj}$ pode ser determinada usando a regra da cadeia:

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial net_k}\frac{\partial net_k}{\partial w_{kj}} \tag{12}$$

Aplicando a equação (11) tem-se:

$$\frac{\partial net_k}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}}\sum w_{kj} o_j = o_j \tag{13}$$

Pode-se então definir:

$$\delta_k = -\frac{\partial E}{\partial net_k} \tag{14}$$

Logo, pode-se escrever:

$$\Delta w_{kj} = \eta \delta_k o_k \tag{15}$$

Novamente para calcular $\delta_k = -\partial E/\partial net_k$ usando a regra da cadeia para expressar a derivada parcial em termos de dois fatores, um fator que expressa a variação do erro em relação a saída o_k e outro que expressa a taxa de variação da saída do nó k em relação a entrada do mesmo nó. Assim:

$$\delta_k = -\frac{\partial E}{\partial net_k} = -\frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial net_k} \tag{16}$$

Estes dois fatores podem ser obtidos da seguinte maneira.

Por definição:

$$\frac{\partial E}{\partial o_k} = -(t_k - o_k) \tag{17}$$

onde t_k é o valor desejado de saída e o_k é a própria saída, e ainda:

$$\frac{\partial o_k}{\partial net_k} = \frac{\partial f_k(net_k)}{\partial net_k} = f'_k(net_k) \tag{18}$$

Até aqui tem-se então que:

$$\delta_k = (t_k - o_k) f'_k(net_k) \tag{19}$$

Obtém-se, desta forma para qualquer neurônio ou nó da camada de saída o seguinte:

$$\Delta w_{kj} = \eta(t_k - o_k) f'_k(net_k) o_j = \eta \delta_k o_{jk} \tag{20}$$

Já no que tange os neurônios das camadas intermediárias, não tem-se explicitamente os valores desejáveis, ou seja, os t_j. Sendo assim, as condições que cercam estes neurônios podem ser reescritas como segue:

$$\Delta w_{\mathrm{kj}} = -\eta \frac{\partial E}{\partial w_{ji}} \tag{21}$$

$$= -\eta \frac{\partial E}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}} \tag{22}$$

$$= -\eta \frac{\partial E}{\partial net_j} o_i \tag{23}$$

$$= \eta \left(-\frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial net_j} \right) o_i \tag{24}$$

$$= \eta \left(-\frac{\partial E}{\partial o_j} \right) f'_j(net_j) o_i \tag{25}$$

Entretanto o fator $\partial E / \partial o_j$, como já foi dito, não pode ser avaliado diretamente. Em contrapartida, pode-se escrevê-lo em termos de quantidades que são conhecidas e outras quantidades que podem ser calculadas, ou seja:

$$-\frac{\partial E}{\partial o_j} = -\sum_k \frac{\partial E}{\partial net_k} \frac{\partial net_k}{\partial o_j} \tag{26}$$

$$= \sum_k \left(-\frac{\partial E}{\partial net_k} \right) \frac{\partial}{\partial o_j} \sum_m w_{km} o_m \tag{27}$$

$$= \sum_k \left(-\frac{\partial E}{\partial net_k} \right) w_{kj} \tag{28}$$

$$-\frac{\partial E}{\partial o_j} = \sum_k \delta_k w_{kj} \tag{29}$$

Neste caso, pode-se observar que:

$$\delta_j = f'_j(net_j) \sum_k \delta_k w_{kj} \tag{30}$$

O valor do delta de um neurônio interno pode ser determinado em termos do delta da camada superior. Desta forma, começando da camada mais alta, ou seja, da camada de saída, pode-se determinar δ_k usando a expressão (19), e

assim, retropropagar o erro no sentido das camadas mais baixas ou de entrada. Considerando que p representa o número do padrão, tem-se:

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} \tag{31}$$

Como já mencionado, para o j-ésimo neurônio da última camada resulta:

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j (net_{pj}) \tag{32}$$

Entretanto, se o neurônio é de uma camada interna, precisamos determinar δ_{pj} em termos dos deltas da camada mais alta com segue:

$$\delta_{pj} = f'_j (net_{pj}) \sum_k \delta_{pk} w_{kj} \tag{33}$$

Considerando que:

$$o_j = \frac{1}{1 + \exp[-(\Sigma_i w_{ji} o_i + \theta_j)]} \tag{34}$$

Então

$$\frac{\partial o_j}{\partial net_j} = o_j (1 - o_j) \tag{35}$$

Logo, os deltas correspondentes podem ser obtidos das seguintes expressões:

$$\delta_{pk} = (t_{pk} - o_{pk}) o_{pk} (1 - o_{pk}) \tag{36}$$

E ainda:

$$\delta_{pj} = o_{pj} (1 - o_{pj}) \sum_k \partial_{pk} w_{kj} \tag{37}$$

A equação (36) se aplica para os neurônios da última camada e a equação (37) para os neurônios das camadas intermediárias.

Deve-se observar que os "thresholds" θ_j são treinados da mesma maneira que um outro peso qualquer, simplesmente considerando que a entrada θ_j sempre tem o valor unitário. Também é importante observar que a derivada parcial $\partial o_j / \partial net_j$ é igual a $o_j(1-o_j)$ e que o máximo é atingido para $o_j = 0.5$ e, desde que $0 \leq o_j \leq 1$, os mínimos são obtidos em zero e um.

Como comentado anteriormente, a função de ativação dada pela equação (34), não pode alcançar valores 0 (zero) e 1 (um) sem que os pesos sejam infinitamente grandes no sentido positivo ou negativo. No entanto, pode-se considerar uma variação entre 0,1 e 0,9 como bastante razoável para a totalidade das aplicações.

No procedimento de treinamento, considera-se que a rede inicia seus pesos com valores randomizados e diferentes entre si. Escolhe-se um padrão e aplica-se como entrada na rede e se avalia o erro final. A partir daí é feito o ajuste necessário dos pesos utilizando um procedimento apropriado, por exemplo, o de propagação reversa ou "backpropagation". No caso do procedimento adotado para propagação reversa, é calculado $\Delta_p w_{ji}$ para todos os w_{ji} da rede para cada particular p. Este processo é repetido para todos os padrões do conjunto de treinamento resultando em Δw_{ji} para cada um dos pesos que compõem a rede. O objetivo de se modificar os valores dos pesos é o de reduzir as discrepâncias entre o valor da saída atual e a saída desejada. Depois de completada a apresentação de todos os padrões do conjunto de treinamento, um conjunto de pesos é obtido e as saídas devem ser avaliadas dentro da maneira conhecida como propagação direta ou "feedforward".

Para o treinamento, pode-se utilizar tanto os erros dos padrões de forma individualizada, mostrado na equação (7), como o erro do sistema mostrado na equação (8).

São vários os detalhes que deve se ter em mente quando da implementação de uma rede. No último caso tratado, a questão de como se escolher o valor de η é bastante relevante. Pode-se imaginar que um η de valor grande corresponde a um treinamento rápido, mais isto também provoca oscilações. Rumelhart, Hinton e Williams sugerem que as equações (20) e (31) sejam modificadas para incluir um termo α denominado de *momentum*. Este último termo é introduzido

para evitar que ocorram oscilações durante o treinamento, embora não elimine totalmente este efeito. A equação pode ser rescrita como segue:

$$\Delta w_{ji}(n+1) = \eta \partial_j o_j + \alpha \Delta w_{ji}(n) \qquad (38)$$

O valor *(n+1)* é usado para indicar o *(n+1)ésimo* passo, e α é uma constante. O segundo termo da expressão é usado para que a variação de w_{ji} do passo *(n+1)* seja na mesma direção da variação empreendida no passo *(n)*. É como se fosse aplicado um coeficiente de inércia ao sistema. O *momentum* atua como um filtro passa-baixa, filtra as rápidas oscilações do erro, evitando desta forma o efeito ziguezague. Examinando a variação do erro do sistema, observa-se que α tende a diminuir as oscilações do erro, mesmo para pequenos valores de η.

A medida que o processo de treinamento vai avançando, o valor do η deve ser gradativamente diminuído para evitar oscilações do erro e também, para se obter erros finais menores. Deve-se lembrar que somente é atingido o mínimo efetivo da função do erro se o valor da taxa de aprendizagem for infinitamente pequeno.

3.3 Algoritmos Genéticos

3.3.1 Introdução

As pesquisas sobre modelos computacionais inteligentes têm, nos últimos anos, se caracterizado pela tendência em buscar inspiração na natureza, onde existe um sem-número de exemplos vivos de processos que podem ser ditos inteligentes. Para cientistas de computação, matemáticos e engenheiros, muitas das soluções que a natureza encontrou para complexos problemas de adaptação fornecem modelos interessantíssimos. Embora não se possa afirmar que tais soluções sejam ótimas, não há a menor dúvida de que os processos naturais, em particular os relacionados diretamente com os seres vivos, sejam bem concebidos e adequados ao nosso mundo e forneceram inspiração para os

paradigmas de computação denominados "Simulated Annealing", Redes Neurais e Computação Evolucionária (CE).

Em termos computacionais, CE incorpora uma das metáforas mais apelativas: CE encara a teoria de evolução Darwiniana como um processo adaptativo de otimização, sugerindo um modelo em que populações de estruturas computacionais evoluem de modo a melhorar, em média, o desempenho geral da população com respeito a um dado problema, ou seja, a "adequabilidade" da população com respeito ao ambiente.

Atualmente, CE engloba um número crescente de paradigmas e métodos, dos quais os mais importantes são os Algoritmos Genéticos (Ags) [3,8,26,30,39], Programação Evolucionária (PE), Estratégias Evolucionárias (EE), Programação Genética (PG), e Sistemas Classificadores (SCs), entre outros [26]. De fato, SCs e PG podem ser vistas como aplicações especiais de AGs, enquanto que PE e EEs foram desenvolvidas independentemente dos AGs, e somente agora começam a interagir. De modo geral, técnicas em CE dão origem aos denominados Algoritmos Evolucionários (AEs).

3.3.2 Descrição do Algoritmo

Algoritmos Genéticos estão entre as mais difundidas e estudadas técnicas de CE, pela sua flexibilidade, relativa simplicidade de implementação, e eficácia em realizar busca global em ambientes adversos.

Busca e Otimização

O objetivo é encontrar a melhor combinação dos fatores, ou seja, a combinação de fatores que proporcione o melhor desempenho possível para o sistema em questão. Em termos técnicos, o conjunto de todas as combinações possíveis para os fatores constitui o denominado *espaço de busca*. Não é difícil perceber que existe uma dualidade entre os conceitos de busca e otimização, de tal modo que todo problema de busca pode ser considerado um problema de otimização e vice-versa [3,8,26,39].

Sem perda de generalidade, consideremos somente problemas de maximização. Dada uma função $f:\Re^n \rightarrow \Re$ e um espaço de busca $\mathbf{S} \subseteq \Re^n$, o problema de otimização pode ser formulado assim:

$$\text{maximizar} \quad f(\chi) \mid \chi \in \boldsymbol{S}. \tag{39}$$

Em termos de um problema de busca, o mesmo problema pode ser escrito:

$$\text{encontrar} \quad \chi^* \mid f(\chi^*) \geq f(\chi), \forall \chi \in \boldsymbol{S} \tag{40}$$

A função $f(.)$ pode ser derivável ou não, uni ou multidimensional, e o espaço de busca **S** pode ser contínuo ou discreto, finito ou infinito, côncavo ou convexo. O problema é denominado unimodal quando há somente um ponto máximo χ^* no espaço de busca, ou multimodal em caso contrário. Além disso, a função $f(.)$ pode ser estacionária ou não-estacionária, isto é, variável com o tempo.

Otimização na Natureza e AGs

Vista de forma global, a evolução natural implementa mecanismos adaptativos de otimização que, embora estejam longe de ser uma forma de busca aleatória, com certeza envolvem aleatoriedade. É esse tipo de busca inteligente, mas não determinística, que os AGs tentam imitar.

Os AGs pertencem a classe dos métodos probabilísticos de busca e otimização, embora não sejam aleatórios. Usa-se o conceito de probabilidade, mas AGs não são estúpidas buscas aleatórias. Pelo contrário, AGs tentam dirigir a busca para regiões do espaço onde é provável a existência dos pontos ótimos.

Fundamentos dos AGs

Uma definição que o autor da referência considera mais plausível de algoritmo genético é a que segue:

Algoritmos Genéticos (AGs) são métodos computacionais de busca baseados nos mecanismos de evolução natural e na genética. Em AGs, uma população de

possíveis soluções para o problema em questão evolui de acordo com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores a medida que o processo evolutivo continua.

Terminologia

A terminologia utilizada em engenharia genética define que:

- Cromossomo: É a completa descrição genética de um indivíduo. É uma coleção de características primitivas denominadas de gene.
- Gene: É a representação simples de uma das característica de um cromossomo.
- Genótipo: É a estrutura genética explícita de um cromossomo.
- Alelo: É o valor específico que o gene pode assumir.

Características Primárias

- AGs operam numa população (conjunto) de pontos, e não a partir de um ponto isolado.
- AGs operam num espaço de soluções codificadas, e não no espaço de busca diretamente.
- AGs necessitam somente de informação sobre o valor de uma função objetivo para cada membro da população, e não requerem derivadas ou qualquer outro tipo de conhecimento.
- AGs usam transições probabilísticas, e não regras determinísticas.

Representação Cromossômica

O primeiro passo para aplicação de AGs a um problema qualquer é representar cada possível solução χ no espaço de busca como uma seqüência de símbolos s gerados a partir de um dado alfabeto finito A. No caso mais simples, usa-se o alfabeto binário A= {0,1}, mas no caso geral tanto o método de representação quanto o alfabeto genético dependem de cada problema.

Usando algumas das metáforas extremamente simplistas, mas empregadas pelos teóricos e praticantes de AGs com freqüência, cada seqüência s corresponde a um cromossomo, e cada elemento de **s** é equivalente a um gene. Como cada gene pode assumir qualquer valor do alfabeto A, cada elemento de A é equivalente a um alelo, ou seja, um valor possível para um dado gene. A posição de um gene num cromossomo, ou seja, o índice dentro da seqüência, corresponde a um "locus gênico".

Além disso, na maior parte dos AGs assume-se que cada indivíduo seja constituído de um único cromossomo, razão pela qual é comum usarem-se os termos indivíduo e cromossomo indistintamente em trabalhos científicos e livros textos. A grande maioria dos AGs propostos na literatura usam uma população de número fixo de indivíduos, com cromossomos também de tamanho constante.

Fluxo Básico

AGs são algoritmos iterativos, e a cada iteração a população é modificada. Cada iteração de um AG é denominada uma geração, embora nem todos os indivíduos de uma população sejam necessariamente "filhos" de indivíduos da população na iteração anterior. Este fluxo segue em primeiro uma inicialização, ou seja, geração da população ou indivíduos, e avaliação destes indivíduos. Após, num processo iterativo, temos o ciclo que corresponde a seleção, recombinação e mutação.

Avaliação e Adequabilidade

AGs necessitam da informação do valor de uma função objetivo para cada membro da população, que deve ser um valor não-negativo. Nos casos mais simples, usa-se justamente o valor da função que se quer maximizar. A função objetivo dá, para cada indivíduo, uma medida de quão bem adaptado ao ambiente ele está, ou seja, quanto maior o valor da função objetivo, maiores são as chances do indivíduo sobreviver no ambiente e reproduzir-se, passando parte

de seu material genético a gerações posteriores. Neste trabalho é utilizado o termo "adequabilidade" como resultado da avaliação.

Seleção

O mecanismo de seleção em AGs emula os processos de reprodução assexuada e seleção natural. Em geral, gera-se uma população temporária de N indivíduos extraídos com probabilidade proporcional à adequabilidade relativa de cada indivíduo na população. Neste processo, indivíduos com baixa adequabilidade tem alta probabilidade de desaparecerem da população.

Recombinação

O processo de combinação é um processo sexuado, ou seja, envolve mais de um indivíduo, havendo troca de fragmentos entre pares de cromossomos, na maioria das vezes através de um processo aleatório. Emula o efeito denominado de "crossover".

Mutação

O processo de mutação em AGs é equivalente a uma busca aleatória. Basicamente, seleciona-se uma posição num cromossomo e muda-se o valor do gene correspondente aleatoriamente para outro alelo possível.

Condições de Término

Como estamos tratando de problemas de otimização, o ideal seria que o algoritmo terminasse assim que o ponto ótimo fosse descoberto. Já no caso de funções multimodais, um ponto ótimo pode ser o suficiente, mas pode haver situações onde todos ou o maior número possível de pontos ótimos sejam desejados. Um problema prático é que, na maioria dos casos de interesse, não se pode afirmar com certeza se um dado ponto ótimo corresponde a um ótimo global. Como conseqüência, normalmente usa-se o critério do número máximo

de gerações ou um tempo limite de processamento para parar um AG. Outro critério plausível é parar o algoritmo usando a idéia de estagnação, ou seja, quando não se observa melhoria da população de várias gerações consecutivas.

Considerações práticas

Num AG básico, o usuário deve definir o tamanho da população, N, além das probabilidades de recombinação e mutação, respectivamente, Prec e Pmut. Em AGs mais sofisticados, há ainda mais parâmetros, comprometendo parte da robustez dos algoritmos. Infelizmente, não há regras claras para a escolha desses parâmetros.

Quanto ao parâmetro N, a intuição indica que "quanto mais, melhor será", uma vez que, em última análise, com uma população infinita cobrindo todo o espaço de busca, a solução ótima seria obtida na primeira geração. Na prática, é óbvio, temos que nos consignar com populações finitas. Partindo da análise de esquemas, Goldberg concluiu que, para cromossomos binários de comprimento l, o tamanho ótimo deve ser uma função exponencial de l [26].

Convergência Prematura

Usando o modelo do AG simples para a otimização de funções multimodais, um fenômeno que se observa comumente é que o AG converge rapidamente (em umas poucas dezenas de gerações) para um ponto de alta qualidade , mas não o ótimo global, num fenômeno denominado convergência prematura.

Por exemplo, supor que na população inicial um dado indivíduo próximo de um ótimo local, mas não global, tem um valor de adequabilidade muito maior que o restante da população. Devido ao processo de seleção, tal super indivíduo terá vários descendentes nas próximas gerações. Em casos extremos, indivíduos próximos ao ótimo global mas com baixa adequabilidade relativa, podem ser completamente extintos, o que geralmente resulta em convergência da população para um ótimo local não global. Assim sendo, convergência prematura está intimamente relacionada com a perda de diversidade da população. Em CE, diversidade é um conceito abstrato que indica o grau em

que as mais diversas regiões do espaço de busca estão representadas na população, ou seja, o quão variada é a população. No Ambiente proposto neste trabalho é permitida a introdução de informação genética nova de maneira a manter o espaço de busca não estritamente localizado.

Problemas Enganadores

Isto ocorre em problemas em que há fragmentos cromossômicos que têm alta adequabilidade em separado mas que, juntos, formam uma solução não ótima. Estes problemas surgem principalmente em cromossomos binários, enganam a busca genética e tendem a fazer o AG convergir para uma solução que não é globalmente ótima, principalmente devido a recombinação. Neste trabalho, principalmente por estar-se utilizando cromossomos não binários, não é executada a fase de recombinação, o que evita o aparecimento deste tipo de problema.

Estratégias Elitistas

Neste modelo, garante-se que os melhores indivíduos de uma geração sempre aparecerão na geração seguinte. Ou seja, se a elite da população corrente não estiver presente na próxima população em decorrência de alguma operação genética, então os elementos ausentes são inseridos artificialmente no lugar dos piores indivíduos. O método mais comum de elitismo supervisiona apenas o melhor indivíduo da população, mas para grandes populações pode ser interessante garantir que, os dez melhores indivíduos sempre estejam presentes na próxima geração.

Reprodução de Estado Estável

Ao invés de substituir toda a população de uma vez, este modelo pressupõe que somente alguns indivíduos devam ser trocados a cada geração [26]. No caso mais simples, insere-se apenas um indivíduo por vez no lugar do pior indivíduo da população atual.

Aplicação de AGs em Redes Neurais Multicamadas

Geralmente o treinamento de redes multicamadas é realizado por algum método de gradiente, quase sempre uma variante da regra delta, usando o algoritmo de propagação reversa ("backpropagation") para o cálculo de derivadas parciais.

Com o objetivo de proporcionar uma busca mais global, AGs podem ser empregados para determinação dos pesos de uma rede multicamadas. Neste caso, cada rede com seus pesos constitui um indivíduo. Os pesos podem ser representados por longas seqüências binárias ou números reais. A avaliação de cada indivíduo é feita com base num conjunto de padrões de teste, de modo que a adequabilidade seja uma função decrescente do erro quadrático.

Programação Evolucionária

Os métodos de Programação Evolucionária (PE) foram originalmente desenvolvidos por Fogel [26]. Tipicamente, em PE há uma população de N indivíduos que são copiados na totalidade numa população temporária e sofrem mutações variáveis.

Um torneio estocástico é então realizado para extrair a seguinte população deste grupo de 2N indivíduos. Não há nenhuma restrição formal que implique que o tamanho da população deva ser constante, e não há recombinação.

A técnica, que no Ambiente está apresentada como algoritmo genético, estaria melhor definida se fosse utilizada como descrição, o título de programação evolucionária (PE), apesar de que os outros paradigmas de CE, incluindo os PEs, geralmente não operam com cromossomos, tipicamente empregado em AGs.

Do ponto de vista purista, não de trata de um AG, porque , segundo Holland e seus discípulos [3,26], recombinação é fundamental.

O autor manteve o título de algoritmo genético somente pelo fato de este estar mais difundido no mundo técno-científico, e considera também, que a

motivação do desenvolvimento do referido algoritmo no Ambiente proposto teve inspiração genética e por consecutivas necessidade de adaptação ao perfil do trabalho a que se destina o Ambiente, suas características acabaram por ser mais pertinentes a PE.

3.4 Coexistência dos Algoritmos

Como já mencionado, o algoritmo de treinamento tipo propagação reversa ou "backpropagation", apesar de ser um algoritmo amplamente utilizado, apresenta certos inconvenientes, pois a solução pode não convergir plenamente para um mínimo desejado caso o espaço de solução seja convoluído. Isto decorre do fato deste algoritmo ser afeto pelos denominados mínimos locais, pois, apesar de ser eficaz em muitos casos, o algoritmo de gradiente somente busca um mínimo local da superfície do erro da rede multicamada [2,3,5,10,15,42,43]. Estes mínimos locais podem retardar e até mesmo paralisar o processo de obtenção da solução global ótima. Este problema pode ser contornado fazendo-se uso de um algoritmo de soluções ótimas tipo algoritmo genético, principalmente na fase inicial do treinamento [3,26,30,39]. O algoritmo genético não é tão afetado pelos mínimos locais pois não envolve minimização de gradiente, entretanto, quando o erro atinge valores relativamente pequenos, a convergência no algoritmo genético se torna um pouco crítica e demorada. No algoritmo genético não existe uma granularidade de evolução do erro definida. O que se quer dizer com isto é que a busca da solução deste algoritmo é feita de uma maneira não exatamente contínua, ou seja, pode decorrer um período onde praticamente o erro não evolui ou evolui de uma forma muito lenta e, numa única iteração seguinte, este erro dar um "mergulho" podendo até acabar atingindo o erro desejado encerrando o processo de treinamento. Por tudo isto, é interessante que se faça uma composição de algoritmos, primeiramente o genético para inicializar os pesos da rede e após o "backpropagation", de maneira a se obter uma solução mais genérica.

Uma forma de se contornar em parte o problema dos mínimos locais no caso do algoritmo de propagação reversa, é provocando-se uma randomização controlada, ou seja, provocar uma alteração de maneira randômica nos pesos o suficiente para se escapar da região do mínimo local toda vez que o processo de convergência do erro deixar de acontecer. Esta solução na maioria das vezes já

se apresenta como suficiente para problemas não complexos e tem como vantagem a relativa simplicidade em relação à utilização de algoritmos genéticos e o "simulated annealing" discutido no parágrafo seguinte. Naturalmente, esta solução não garante que não se possa cair novamente num outro mínimo local, sendo que, deve-se redisparar o processo de randomização. No Ambiente, foi definido um número máximo de randomizações automáticas, de modo que, a partir de alguns ciclos, o processo de treinamento possa ser encerrado caso a solução ótima não tenha sido atingida.

O método proposto de randomização controlada é uma derivação do método "simulated annealing" [3][24], apesar de não usar o método de regressão tradicional. Este método é um modelo de processo estocástico baseado em um princípio utilizado em metalurgia, que corresponde a elevar a temperatura de um material até a temperatura de fusão deste material, e após, esfriá-lo lentamente numa forma escalonada. Se este esfriamento é lento suficiente, o material atinge um baixo equilíbrio de energia. Este decréscimo lento da energia (randômico) das partículas do material leva a um mínimo estado de energia. Em outras palavras, quando os átomos de uma peça de metal estão alinhados randomicamente, este metal é frágil e quebradiço. No processo de recozimento, o metal é aquecido a uma alta temperatura, causando violentos choques entre os átomos. Se este metal é esfriado rapidamente, a micro estrutura deste material é mantida num estado randômico instável. Ao contrário, se a temperatura é abaixada lentamente, os átomos tendem a se manter dentro de padrões relativamente estáveis para aquela temperatura.

Este mesmo processo é aplicado na resolução de equações. "Simulated annealing" pode realizar uma otimização pela perturbação randômica de variáveis independentes (pesos no caso das redes neurais) e caminhar no sentido da obtenção do menor erro da função. Neste algoritmo, o sistema se move de um estado para outro de acordo com um método estocástico de seleção. A nova energia neste estado é avaliada. Se a energia é menor que a do estado anterior, então este novo estado é aceito. O que se objetiva é alcançar o ponto de mínima energia ou o estado de aterramento. Um estado de energia alto, no entanto, não necessariamente representa um estado que deve ser rejeitado. Similarmente a um mínimo local, o único caminho para se escapar, é aumentando a energia, ainda que temporariamente. Neste caso, serão aceitos novos estados. Baseado numa teoria randômica, o sistema eventualmente atingirá um mínimo

mensurável. Entretanto, não há uma maneira segura de se estabelecer que foi localizado o mínimo do sistema. Uma abordagem probabilística baseado em sistemas físicos é usada de forma a otimizar este procedimento [24]. O que se pressupõe é que partindo de qualquer temperatura, o sistema pode atingir um ponto de quase equilíbrio. Se é possível obter um estado de quase equilíbrio a uma alta temperatura e a temperatura é vagarosamente diminuída, vagarosamente o suficiente para manter o quase equilíbrio, o sistema converge para o estado de aterramento.

O termo temperatura corresponde a um desvio padrão dado por um gerador de números randômicos.

Observar que no "simulated annealing" é necessário se preestabelecer um conjunto de duplas, temperatura inferior/temperatura superior. Isto obriga o usuário do método ter um conhecimento suficiente para que possa fazer uma estimativa coerente com a necessidade do processo tratado. O número total de tentativas é neste método igual ao produto do número de temperaturas pelo número de iterações em cada temperatura. Em aplicações do mundo prático, é muito difícil logo de início se poder estimar qual é a superfície do erro do problema analisado, para que se possa fazer com que o próprio algoritmo faça a rejeição de um mínimo local por exemplo. No processo de randomização utilizado no Ambiente descrito neste trabalho, esta randomização ocorre de uma forma um tanto mais intuitiva, o que facilita a utilização do método com a segurança de que se está utilizando-o de uma forma satisfatória, sem exigir um esforço adicional do usuário.

Neste particular, a randomização ocorre baseada na distância euclidiana do erro atual com o erro mínimo especificado pelo usuário. Quando o erro atual cessa de convergir, é provocada uma randomização em todos os pesos da rede, ou seja, é introduzido um ruído sobre todos estes pesos . A amplitude deste ruído é dada pela distância do erro atual e o erro mínimo especificado. Desta forma, cada peso sofrerá uma variação diferenciada entre si, mas não o suficiente para descaracterizar todo o treinamento feito até então.

Este processo de randomização atua conjuntamente com o ajuste automático da taxa de aprendizagem, também discutida ao longo deste trabalho. Quando a convergência deixa de ocorrer, é manipulado o valor instantâneo da taxa de

aprendizagem, reduzindo este valor, até que se estabeleça novamente a convergência da função do erro. A randomização ocorrerá sempre que a taxa de aprendizagem atingir um valor predefinível suficientemente pequeno de maneira que não ocorra mais um aprendizado adequado, ou seja, após sucessivas iterações o erro permanece aproximadamente o mesmo.

A forma de randomização apresentada facilita em muito o trabalho do usuário do método, visto que este necessita somente definir o número máximo de randomizações que devem ser aceitas e o limite mínimo da taxa de aprendizagem antes de ocorrer uma randomização automática. Valores que sugerimos é um número de até cinco randomizações e a taxa de aprendizagem não inferior a 0,000001. Estes valores proporcionam resultados adequados a praticamente totalidade das aplicações.

O número de iterações necessárias para que cada um dos algoritmos apresentem a sua melhor solução é sensivelmente menor no algoritmo genético. Entretanto, o tempo computacional gasto a cada iteração neste mesmo algoritmo é também sensivelmente maior. Desta forma se estabelece uma solução de compromisso entre cada um dos algoritmos tratados, sendo então que o próprio usuário deve definir o número de iterações a partir do que haverá a migração de um algoritmo para o outro.

No algoritmo "backpropagation" é sempre garantido que se atinja a um mínimo da superfície da função do erro para aquela região do espaço de busca, mas isto de fato só ocorre se a mudança dos pesos é infinitesimal. A magnitude de variação dos pesos, ou amplitude do passo, é determinado como já mencionado, pelo valor da taxa de aprendizagem η. Na prática, o que se deseja é que o valor desta taxa seja o maior possível de modo a fazer com que a convergência ocorra de uma forma mais rápida possível. Se o passo for grande, entretanto, o erro do sistema começa a oscilar, nunca atingindo um mínimo. Neste algoritmo, somente usa-se a primeira derivada parcial da superfície do erro na localização atual dentro do espaço de pesos. Para determinar um tamanho apropriado do passo, pode ser usado algum conhecimento que se tem sobre a curvatura da superfície do erro, ou seja, a derivada parcial de segunda ordem do corrente ponto dentro do espaço de pesos. Algumas técnicas são apresentadas em [16] com o propósito de estimar o valor, ou uma aproximação, da derivada segunda.

O que ocorre é que, em geral, estes métodos são caros do ponto de vista computacional em relação ao próprio algoritmo "backpropagation".

Em [40, 41] são descritos vários mecanismos que aceleram o tempo de convergência destacando as técnicas ditas adaptativas locais que se baseiam na variação de cada peso de forma isolada. Em [40] são descritos os métodos Delta-Bar-Delta, SuperSAB, Gradiente Conjugado, Quickprop e Rprop.

No Ambiente proposto neste trabalho, é utilizada uma forma simplificada de ajuste automático da taxa de aprendizado e extremamente eficiente correspondendo ao objetivo proposto, ou seja, a praticidade e eficiência com baixo custo computacional. O método empregado consiste em se escolher inicialmente um valor relativamente alto para a taxa de aprendizagem, valor algumas vezes a unidade (tipicamente menor que 3). O valor da taxa de aprendizagem é reduzido por um passo predefinido toda vez que o erro resultante do treinamento não decrementar. Este método apresenta como vantagem o baixo custo computacional basicamente resultante de uma única comparação entre o erro do instante atual em relação ao do instante anterior. Nos métodos apresentados em [40], é necessário obtenção de derivadas, determinação de parâmetros de ajuste empírico, que de uma maneira geral tem um desempenho satisfatório caso os parâmetros de decremento/incremento dos valores dos pesos, bem como os limites máximos forem corretamente estimados.

3.5 Aspectos Gerais do Treinamento

3.5.1 O Problema dos Mínimos Locais

Um problema relativo ao treinamento está relacionado com os denominados mínimos locais, efeito bastante perceptível quando a superfície da função do erro é bastante convoluída.

Este efeito pode ser perfeitamente observado na figura 5. Nesta figura pode-se notar que a função representada apresenta um mínimo global (min) e um

mínimo local (local). Ocorrem ainda situações onde o erro praticamente se mantém constante (estacionário) mesmo para uma variação considerável no ajuste dos pesos. Observa-se que mesmo após um certo número de iterações o erro permanece estático.

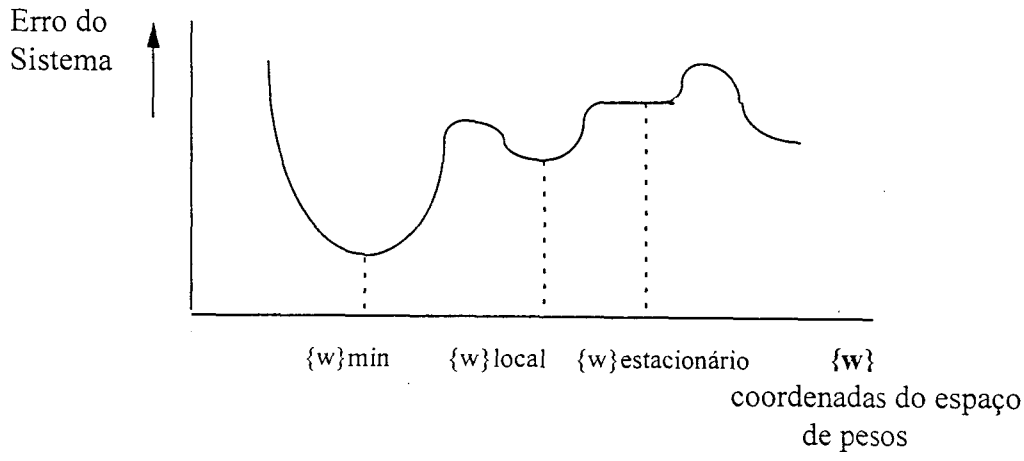


Figura 5 - Superfície da função do erro

O que se espera sempre durante o treinamento propriamente dito, é que seja atingido o mínimo global de uma maneira mais rápida possível.

Vale lembrar que um mínimo local nem sempre se apresenta como algo danoso. Deve-se sempre ter em mente que a rede neural é basicamente um aproximador de soluções, ou seja, a solução final apresentada possue um erro sempre associado. O importante é que ao término do treinamento se consiga atingir o erro final desejado em termos de cada padrão e/ou em termos do erro do conjunto total de padrões de treinamento. O que podemos dizer é se, o resultado esperado foi obtido fora do mínimo global, é muito provável então que, com um treinamento mais apurado, seja até possível obter o mesmo resultado dentre uma topologia de rede mais econômica por assim dizer, ou seja, com uma rede com menor número de neurônios ou até mesmo camadas de acordo com o caso. De uma maneira geral, o que deve balizar o fim de um processo de treinamento são os testes executados a partir de um conjunto de padrões escolhidos especialmente para esta etapa, em outras palavras, o desempenho de uma rede após o treinamento só será seguramente determinado após se ensaiar efetivamente a rede com padrões de teste apropriados.

Algoritmos como o genético [3] e o "simulated annealing" [3][24] permitem atenuar sensivelmente o problema dos mínimos locais bastante comuns nos algoritmos de gradiente descendente.

Deve-se observar que os padrões utilizados no treinamento devem ser evidentemente representativos do problema analisado. Se após encerrado o treinamento e durante os ensaios os resultados não forem satisfatórios, deve-se preocupar em reavaliar o padrão de treinamento original. Quando o conjunto de padrões de treinamento e o conjunto de padrões de teste estão bem caracterizados, é possível realizar uma mesclagem entre eles e observar o novo comportamento após a rede ter sido novamente treinada. De um modo geral, quando o treinamento não corresponde ao esperado, pode-se dizer que a função matemática que representa o problema não está dominada nem mesmo em parte, ou seja, nem a quantidade de imperfeições ou convoluções existentes se conhece.

Caso esta situação se apresente de uma forma complexa, uma maneira de contorná-la é dividindo o problema em pequenos subproblemas, ou seja, buscar especializar uma rede para cada trecho crítico do problema observado. Com isto se consegue redes efetivamente menores favorecendo sem dúvida o desempenho final, fato de suma importância quando trabalha-se com redes em tempo real. Desta forma, o que se busca não é obter uma única rede complexa que atenda toda a problemática do processo envolvida, e sim, várias pequenas redes, bem mais simples cada uma especializada em uma parte do problema. O critério de seleção de qual das redes deve ser selecionada num determinado instante é feito pelo usuário. No Ambiente descrito neste trabalho foi escolhido o valor instantâneo da referência com o fator de seleção de qual das redes deve ser ativada a cada instante. Este critério é escolhido por se tratar de um ambiente especializado para realização de controladores.

Outro aspecto interessante de ser observado durante a fase de treinamento é que uma rede não escalona bem um problema, ou seja, quando se aumenta o tamanho de uma rede, o tempo de treinamento não aumenta na mesma proporção. Durante os ensaios, ocorreram situações onde o tempo de treinamento diminuiu quando foram inseridos mais neurônios numa determinada rede. Apesar disto não ser uma regra, a alteração no tempo de treinamento se deve antes de mais nada a complexidade do problema

apresentado à rede. Deve-se no entanto observar que esta situação acontece quando se define um erro mínimo desejado, a partir de um valor predefinido, diferente do mínimo da função. Se a opção é esgotar o treinamento, ou seja, obter o valor mínimo de erro possível, a observação do tempo já não ocorre, visto que, se uma rede é maior em termos de sua topologia que uma outra, a que tiver mais neurônios tende a apresentar um erro final inferior a outra rede, o que leva a um número menor de iterações necessárias para o mesmo valor de erro, pois pode-se dizer, que a capacidade de aprendizado é maior.

3.5.2 Validação

É realmente temerário treinar uma rede e colocá-la imediatamente para realizar um serviço. É mais prudente avaliá-la primeiro. Este processo é denominado de *validação*. O procedimento mais típico é separar as situações conhecidas em dois conjuntos disjuntos. Um conjunto é usado para o treinamento e outro usado como conjunto de validação, ou seja, usado efetivamente para testar a rede. Não deve-se subestimar a validação. Esta etapa e tão ou até mesmo mais importante que o próprio treinamento [3].

Duas situações inconvenientes podem acontecer. Quando se finda uma operação de treinamento, se espera que o erro apresentado pela rede durante a validação seja próximo ao erro obtido durante o treinamento.

Se isto não ocorrer, e a diferença é grande, deve-se suspeitar que os dois conjuntos, o de treinamento e o de teste não representam o mesmo tipo de população. A segunda situação que pode ter acontecido é o sobretreinamento.

3.5.3 Sobretreinamento

Ocorre situações onde após a rede estar garantidamente treinada, apresentando um erro final relativamente baixo, quando se excita a rede utilizando-se o padrão de teste, os resultados se apresentarem de uma forma geral inadequados. Esta situação pode ainda mais se agravar quando se tem poucos vetores no padrão de treinamento utilizado relativamente ao tamanho da rede.

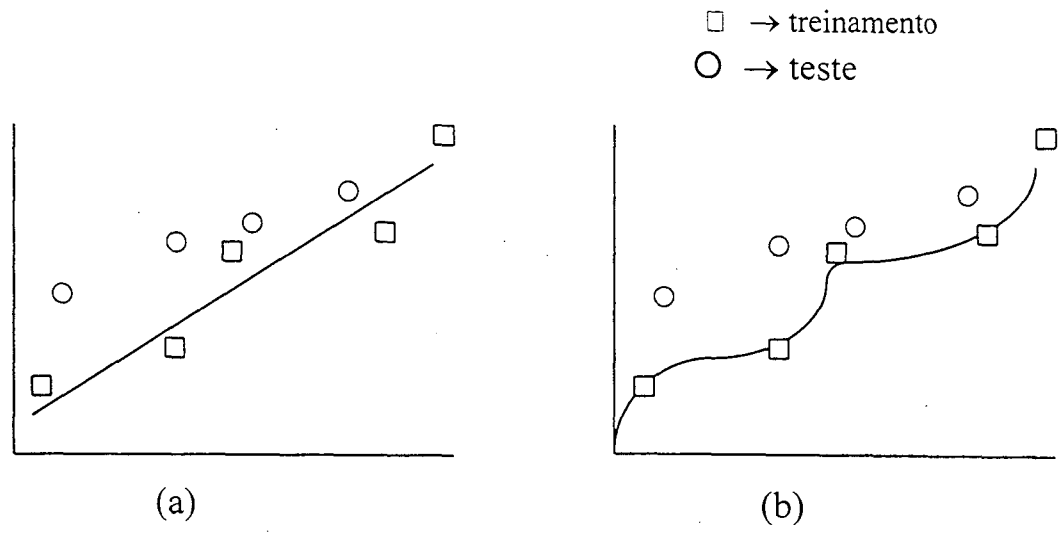


Figura 6 - Treinamento & teste

O problema do sobretreinamento pode ser melhor descrito na figura 6. Tanto na figura 6(a) como na figura 6(b) temos representados por quadrados os pontos que simbolizam cada um dos vetores de dados utilizados no padrão de treinamento utilizado e, por círculos, os vetores de dados utilizados durante os testes. Na figura 6(b) temos um exemplo de uma rede que foi sobretreinada. O erro efetivo final alcançado durante o treinamento na figura 6(b) é muitas vezes menor que o erro final obtido na figura 6(a), no entanto, o resultado conseguido a partir da figura 6(a) é mais interessante, pois apresenta um comportamento mais perfeitamente definido, sem duvida mais interessante do ponto de vista de um controlador.

Numa rede sobretreinada não pode-se ter segurança com relação ao caminho que a função descreve para unir os pontos. Quando, da mesma forma, utiliza-se uma topologia de rede (número de neurônios & números de camadas) maior que a necessária, aumenta sensivelmente o risco do sobretreinamento. É decepcionante encerrar um treinamento com um erro final extremamente baixo e constatar que os resultados durante a fase de teste não correspondem em nada a expectativa gerada durante o treinamento, pois é natural, que após ter-se obtido um erro insignificante durante o treinamento se pressuponha que os testes deverão ser melhores ainda.

Nunca deve-se desprezar a capacidade de aprendizado de uma rede com uma única camada de três neurônios.

Tanto mais estaremos sujeitos a erros sutis como os acima mencionados quanto mais ciclos treinamento/teste forem feitos. A princípio, a cada início de treinamento, o conjunto de pesos estão randomizados numa diferente situação, e pode, a cada ciclo de treinamento se obter um conjunto de pesos distintos entre si. No Ambiente aqui apresentado, é possível iniciar um treinamento a partir de um conjunto de pesos já existentes, ou seja, continuar o treinamento a partir do ponto onde se havia parado anteriormente.

Outra política interessante de ser adotada é que mesmo encerrada a fase de treinamento e validação com sucesso, se repetir o ciclo de modo a procurar se obter uma outra rede e depois comparar o desempenho da segunda com a obtida anteriormente. Isto se deve ao fato de que uma rede mesmo treinada e testada sobre as mesmas condições, tem um comportamento ligeiramente diferente da outra, pois, é praticamente impossível que ambas apresentem exatamente o mesmo conjunto de pesos.

Em algumas situações se observa uma relativa dificuldade de se realizar o treinamento sob determinadas condições. Pode ser isto um alerta de que a randomização provocada inicialmente nos pesos da rede não está muito satisfatória, podendo estar acontecendo que a randomização está mantendo os valores dos pesos próximos uns dos outros ou exatamente o oposto, muito distantes. Isto demonstra a importância que o gerador de números randômicos representa durante o processo de treinamento.

Uma rede após treinada, como funciona é uma tarefa complexa de entender. Técnicas estritamente matemáticas de verificação do desempenho de uma rede estão ainda numa fase preliminar. Muitos pesquisadores têm dedicados seus esforços no sentido de buscar respostas para esta questão, mas até então os resultados tem sido mais de interesse teórico do que exatamente práticos [2,3].

3.5.4 Entendendo o Papel do Neurônio

Basicamente, uma rede conexionista deve ser treinada para discriminar determinados padrões de modo a classificá-los. É natural que se faça a pergunta

de quantas camadas e quantos neurônios em cada camada são necessários para que a uma determinada rede possa realizar esta tarefa.

É realmente impraticável definir uma arquitetura de uma rede dado a especificação do problema [2]. Vamos procurar entender melhor o papel de cada neurônio no conjunto de uma rede. Cada neurônio da primeira camada forma um hiperplano dentro do espaço de padrões, pois os neurônios de entrada realizam uma soma linear destas entradas. Como pode-se ter um diferente número de entradas, esta soma linear pode representar uma linha num espaço bi-dimensional, um plano no espaço tri-dimensional e, hiperplanos num espaço de dimensão N.

Este raciocínio nos leva a pensar em termos de particionar uma diferente população dentro de pequenas hiper-regiões. Uma vez que tenha-se realizado esta partição, pode-se então classificar as várias regiões propriamente.

Isto indica que cada hiper-região necessita de $2N$ neurônios na primeira camada intermediária, um neurônio para cada um dos lados da hiper-região. Na camada seguinte, um neurônio é necessário para realizar uma operação AND sobre o conjunto de hiperplanos. Sendo assim, visto que cada neurônio da primeira camada interna forma um hiperplano, um neurônio da segunda camada intermediária forma uma hiper-região a partir das saídas dos neurônios da primeira camada. As saídas dos neurônios da segunda camada só são ativadas se o padrão está dentro desta hiper-região. Esta situação é ilustrada na figura 7.

Nesta figura, considerando o caso onde padrões de uma simples classe c_i estão dentro de várias regiões desconectadas dentro de um espaço de padrões de dimensão N. Cada região semelhante é envolvida por um neurônio da segunda camada. Se um nó da camada de saída puder ser preparado para realizar uma operação OR com todos os c_i neurônios da segunda camada, então qualquer c_i padrão será reconhecido e classificado corretamente independentemente de qual região esteja localizado. A princípio pode-se implementar a função OR arranjando os pesos para serem iguais e menores que um e o threshold do neurônio de saída ser suficientemente baixo, de forma que, qualquer um dos neurônios conectados da segunda camada for para "alto", o neurônio de saída será disparado.

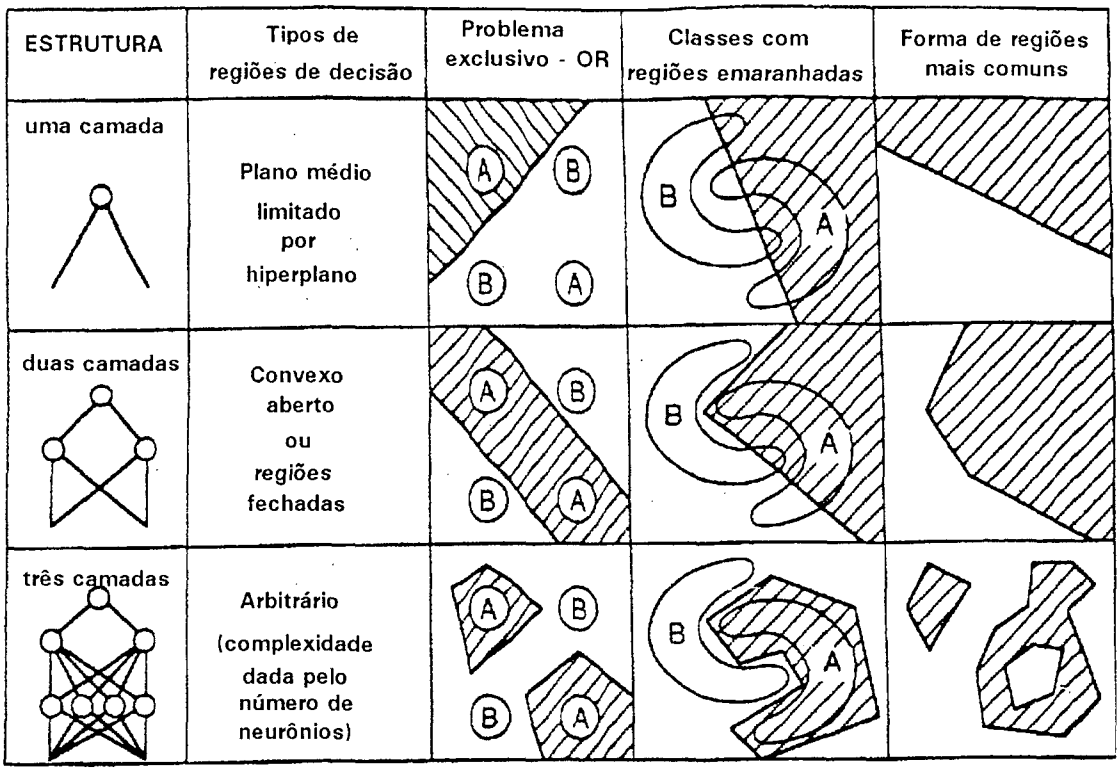


Figura 7 - Papel de cada neurônio

Em princípio, a hiper-região não precisa ser convexa ou ter outro formato complexo. Na prática, as situações não são sempre simples como estão aqui descritas. É muito freqüente um alto grau de redundância de hiperplanos.

4. Redes Neurais Aplicadas à Área de Controle

Nas aplicações voltadas à área de controle de processos a rede tipo multicamadas é a que tem despertado um maior interesse dos pesquisadores [1,5,7,9,10,14,15,18,21,22,23,24,31,33,34,37].

Como já vimos, nestas redes os neurônios estão totalmente interligados com os neurônios das camadas adjacentes. Neste tipo de topologia, a rede é constituída de uma camada de entrada, uma ou mais camadas intermediárias e uma camada de saída.

Os neurônios das camadas internas realizam a modelagem de funções não-lineares, servem também como supressores de ruído e para eliminar pequenos desvios nas entradas.

As saídas são computadas da entrada para saída e o ajuste do peso é feito através de algum algoritmo específico de treinamento. Neste trabalho foram utilizados dois algoritmos distintos, que são: O algoritmo genético e o de propagação reversa também denominado de "backpropagation".

Além da facilidade que as redes neurais têm em representar um controlador para processos complexos ou que apresentem alto índice de não-linearidade, associa-se a isto o fato destas redes terem uma alta imunidade a ruídos e a facilidade de se implementar controladores com estrutura adaptativa.

Técnicas de controle adaptativo têm sido desenvolvidas basicamente para processos que operam sob condições inesperadas ou até mesmo, difíceis de serem previstas, e portanto, impossíveis de serem levadas em conta na construção de um modelo representativo.

A utilização dos controladores neurais tem se demostrado interessante nos casos em que existe uma grande dificuldade na obtenção do modelo matemático do processo envolvido, muito comum durante as implementações práticas.

É importante salientar a necessidade da normalização das entradas e saídas da rede visto que cada neurônio somente pode receber valores entre 0 e 1.

No que tange ao comportamento dinâmico associado ao processo controlado, nas redes neurais não são necessários os sinais de derivada da trajetória desejada. O modelo dinâmico do processo pode ser assimilado pela rede somente considerando os sinais atrasados no tempo de todos os vetores de entrada.

A razão disto é que a camada de entrada da rede funciona como uma espécie de derivador numérico, como pode ser visto na figura 8.

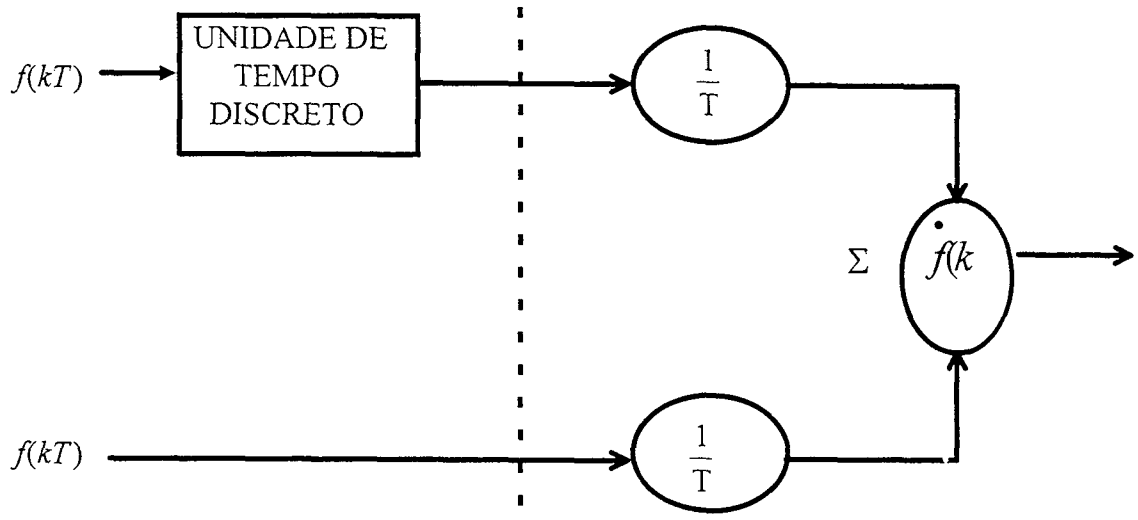


Figura 8 - Derivador numérico

Para que se realize o derivador numérico, é necessário que a entrada atrasada no tempo seja subtraída da entrada atual. Isto só é possível devido ao ajuste interno dos próprios pesos da rede.

Desta forma tem-se a representação aproximada dada pela equação (41), equação típica de um derivador numérico.

$$\dot{f}(kT)= \frac{-f(kT-T)+f(kT)}{T}$$
$$\dot{f}(kT) = \frac{f(kT)-f[(k-1)T]}{T} \qquad (41)$$

De uma forma prática e impírica, se associa o número de atrasos com a ordem do modelo matemático do processo, ou seja, para um modelo de primeira ordem se utiliza somente um atraso *f[(k-1)T]*. Já para um modelo de segunda ordem tem-se dois atrasos, *f[(k-1)T]* e *f[(k-2)T]* e, assim, sucessivamente.

Durante o desenvolvimento deste trabalho foram analisados outros tipos de redes neurais com o intuito de avaliar a vantagem real em relação a rede tipo multicamadas "feedforward".

Em [27,28] é apresentado um método baseado em uma rede neural com memória, que possue uma série de realimentações internas. O propósito é evitar a utilização dos atrasos de entrada e da saída na estimação da ordem do processo. Para isto, a cada neurônio da rede como um todo, é adicionado mais um outro, que funciona como um neurônio auxiliar ao neurônio convencional. Comparando, tem-se notadamente uma diminuição da velocidade de treinamento, pois, na maioria das aplicações práticas, como no estudo de caso do capítulo 6, para uma rede com 2 camadas intermediárias com 3 neurônios em cada camada, teremos neurônios internos em número superior a 7, somando-se a cada um destes, o neurônio auxiliar. Usando-se o método de atraso das entradas, para a captura do efeito dinâmico do processo, tem-se 1, 2 ou 3 neurônios, visto ter-se somente uma entrada. Para a referência não é necessário computar atrasos. A estimação da ordem do processo tipicamente não se apresenta como algo dificultoso, pois, por observação do processo, pode-se estimar a ordem deste e, mesmo por tentativas, tem-se que explorar 2 talvez 3 alternativas. O ponto central é que, deve-se primar pelo tempo final de treinamento, principalmente com respeito ao aprendizado on-line. Já durante o projeto do controlador este tempo não é tão crítico.

Em [2,3,28] é discutida a utilização de redes neurais funcionais ("functional-link net"). Esta rede se apresenta como uma simplificação em relação à rede multicamadas "feedfoward" também conhecida como perceptron multicamadas, por necessitar somente da camada de entrada e de saída e não necessitar de camadas intermediárias. Entretanto, utiliza a regra delta padrão para o

treinamento. A camada de entrada é expandida, sendo que para cada entrada, é utilizado uma combinação de funções ortogonais para se produzir o mapeamento desejado. São utilizadas as funções de base ortonormais do seno e cosseno. Não há outra maneira senão a tentativa e erro para a escolha das funções ortogonais. Pode-se ter funções do tipo sen(πx), cos(πx), sen(2πx), ..., cos(3πx), etc, de acordo com a complexidade do problema. Desta forma, a cada entrada são adicionadas 2 a 3 neurônios. Em [2] é apresentado um problema de paridade-3 onde a rede funcional necessita de 8 neurônios e a multicamada de apenas 7. Já em [28] é apresentado um controlador para um processo de segunda ordem que necessita de 16 neurônios no total. Usando o perceptron multicamadas tem-se algo como 17 a 18 neurônios para o mesmo problema. Em contrapartida, necessita-se gerar os sinais baseados na função seno e cosseno, que nem sempre têm uma solução trivial, já que, boa parte dos transdutores usados em processos industriais possuem saída CC, o que ocorre com o estudo de caso apresentado no capítulo 6.

Se discute atualmente a utilização de redes plásticas (não-paramétricas), denominadas redes polinomiais, visto que o neurônio é baseado num polinômio de segundo grau [45]. Esta rede é capaz de fazer uma seleção das entradas automaticamente. Operando exclusivamente sozinha apresenta incovenientes por não ser capaz de realizar retropropagação do erro, necessário para o controle indireto, não é fácil estabelecer a ordem da função de ativação, e a função quadrática dos polinômios não é adequada para resolver todos os problemas. Desta forma, esta rede é dita ser menos genérica do que a rede perceptron multicamadas. Nos casos práticos, para a grande maioria dos problemas, não existe dificuldade de seleção de entradas. Em [45] é apresentado um modelo que une a rede funcional e a rede polinomial, no entanto não apresenta resultados práticos para serem analisados. Tem como vantagem ser um modelo não-paramétrico (arquitetura da rede é automaticamente definida), no entanto, apresenta as mesmas dificuldades descritas no parágrafo anterior.

Ainda, no que tange a rede funcional, uma questão sobre a plausibilidade biológica deve ser feita. Atualmente existe uma tendência a se questionar a utilização de funções não-monotônicas em redes neurais, já que nos processos biológicos tais funções não ocorrem.

Tudo isto nos leva a concluir que apesar das várias propostas de redes neurais para aplicações em controle que surgiram nos últimos anos, ainda a rede multicamada "feedforward" se apresenta como uma das melhores soluções existentes.

5. Projeto e Implementação de Redes Neurais Multicamadas

5.1 Projetando-se uma Rede Neural Multicamadas

A capacidade de uma rede de se ajustar a um problema está intimamente ligado à arquitetura ou topologia escolhida. Numa rede "feedforward" multicamadas, a escolha adequada do número de neurônios e de camadas intermediárias pode representar a diferença entre o sucesso e o fracasso. Até hoje não foi encontrada uma fórmula que determinasse qual deve ser o tamanho da rede para um dado problema. Entretanto, algumas regras podem auxiliar na determinação das características para um determinado modelo de rede, quanto ao número de camadas intermediárias, quanto ao número de neurônios por camada e quanto ao treinamento [3].

Número de camadas intermediárias

Usar o menor número possível de camadas intermediárias. Começar utilizando sempre uma só camada intermediária. A princípio não há uma razão teórica para se utilizar mais que duas camadas intermediárias. Problemas que requerem mais que duas camadas intermediárias raramente são encontrados na vida real [3]. Na grande maioria das vezes a dificuldade do treinamento está na escolha não adequada dos vetores de treinamento. Observar que o super dimensionamento de uma rede pode provocar o problema do sobretreinamento, e sem dúvida aumenta significativamente o número de mínimos locais, resultando muitas vezes em perdas de tempo durante a fase de treinamento. Em testes práticos nem sempre é possível garantir que uma rede maior leve mais tempo para aprender. Ocorre que uma rede maior tem conseqüentemente uma capacidade de aprendizado também maior, e em certas circunstâncias pode compensar o

tempo adicional de treinamento que seria gasto nesta rede de maiores dimensões. Numa forma geral, quanto maior a rede utilizada tanto menor será o erro final obtido. O importante é definir bem o problema, ou seja, qual é o erro final desejado. Sempre deve-se ter em mente que o erro significativo é o erro obtido durante os testes e não durante o treinamento. Isto quer dizer que quando aumentamos demais o tamanho de uma rede, tipicamente o erro de treinamento diminui, mais o erro do teste, a partir de um ponto, começa a aumentar devido ao problema do sobretreinamento, como mostrado na figura 9. Nesta situação é conveniente diminuir o número de neurônios e usar mais de uma camada intermediária. De preferência manter um número decrescente de neurônios da primeira camada intermediária para a última.

Uma consideração relacionada ao tamanho deve ser feita quando a rede já está parcialmente treinada, ou quando tem-se algum critério de adaptabilidade envolvido. Neste caso, o que se busca é tempo efetivo de treinamento, ou seja, é ideal que se tenha uma rede, a menor possível, e que garanta a princípio um treinamento rápido. De uma forma geral, o critério final de dimensionamento é algo experimental e impírico. Só após os testes é que pode-se concluir efetivamente se a rede está ou não adequada para o problema proposto.

Número de Neurônios

Usar um pequeno número de neurônios nas camadas intermediárias. Um número excessivo de neurônios numa determinada camada é a principal razão do sobretreinamento. A capacidade de processamento da rede aumenta, fazendo com que esta acabe por aprender aspectos insignificantes dos vetores de treinamento. Em contrapartida, usando poucos neurônios, está se enfraquecendo a capacidade da rede em resolver o problema. Em experiências práticas tem-se observado que é usual utilizar um número de neurônios superior a três, ou então, correlacionar os neurônios das camadas intermediárias proporcionalmente ao número de neurônios da camada de entrada. Pode-se utilizar a *regra da pirâmide geométrica* [3], para os casos onde o número de entradas seja relativamente grande. Esta regra diz que o número de neurônios deve decrescer geometricamente da camada de entrada para a camada de saída. Para uma rede com uma única camada intermediária o número de neurônios pode ser dado pela equação:

$$hidden = \sqrt{nm} \tag{42}$$

onde n é o número de entradas e m é o número de saídas. Em [3] é apresentada uma extensão deste critério para mais de uma camada intermediária. Considera-se que o critério da pirâmide geométrica deve ser aplicado exceptuando-se a camada de entrada, ou seja, deve ser iniciado a partir da primeira camada intermediária. Desta forma, somente usando-se a equação descrita acima atinge-se a quase totalidade dos problemas da área de controle, ou seja, uma rede com até duas camadas intermediárias. Por experimentação, o autor sugere que seja utilizado na primeira camada intermediária um número de neurônios igual ou superior ao número de neurônios da camada de entrada.

Para o Ambiente proposto, caso se deseje aumentar o número de neurônios de uma determinada camada, mantendo as demais e continuar aproveitando o treinamento anterior, deve-se manter os pesos anteriormente obtidos e editar diretamente no arquivo de pesos valores randomizados próximos a zero na posição referente à camada onde são introduzidos os novos neurônios e reinicializar o treinamento deste ponto. Se for simplesmente selecionado um número maior de neurônios, são integralmente utilizados somente os neurônios anteriores ao que foi introduzido. Os neurônios após este ponto tem seus valores alterados, ou seja, se for introduzido um neurônio na camada intermediária, até esta camada são mantidos os valores dos neurônios anteriores. A partir desta camada em direção à camada de saída, os valores dos neurônios são perdidos. A edição dos novos valores de peso pode ser feita utilizando-se um editor de texto qualquer que não introduza caracteres de controle adicionais. Em função da robustez dos algoritmos de treinamento, sugere-se que simplesmente seja redisparado outro treinamento. Do ponto de vista prático, o processo de retreinamento normalmente leva menos tempo do que muitas vezes se buscar um editor de texto e alterar diretamente o arquivo de dados dos pesos.

Treinamento

A premissa é treinar o máximo possível. Entretanto, um grande risco que se corre quando se treina em excesso uma determinada rede é o sobretreinamento. A figura 9 exemplifica este problema:

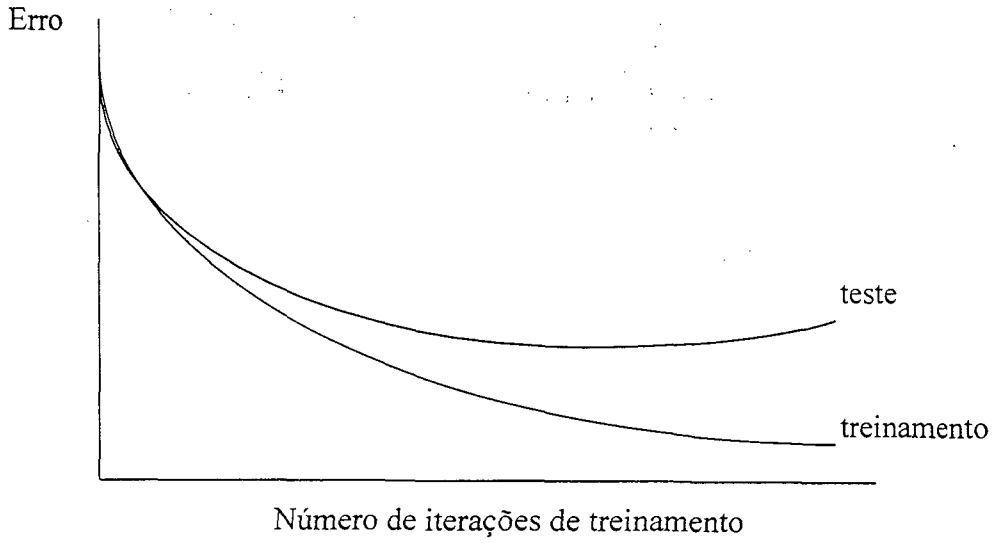


Figura 9 - Erro treinamento & teste

Pela figura, pode-se observar que existe um número ótimo de iterações a partir do qual, mesmo tendo-se um decréscimo contínuo no erro de treinamento, o resultado dos testes não mantém a mesma tendência. Deve-se considerar este problema sempre associado com uma rede sobredimensionada, ou melhor, uma conjunção dos dois fatos, rede sobredimensionada e sobretreinada. Como já foi comentado ao longo deste trabalho, a rede sobredimensionada faz com que a capacidade de processamento seja tal que acabe por aprender aspectos insignificantes do conjunto de vetores de treinamento. Naturalmente que este efeito está intimamente ligado com o treinamento aplicado à esta rede e também com a qualidade ou escolha dos vetores utilizados durante o treinamento.

Acontecendo situação semelhante à da figura acima, algumas considerações devem ser feitas: se isto ocorrer para um determinado conjunto fixo de vetores de teste e treinamento, deve-se procurar mesclá-los, ou seja, é possível que a informação contida em um dos conjuntos não seja representativa do outro conjunto. Esta situação não ocorre no Ambiente proposto, pois os vetores utilizados durante o treinamento são sempre representativos em relação aos

testes. Pode acontecer que o comportamento do processo não tenha sido perfeitamente capturado pelos vetores de treinamento, ou seja, pode ser que o conjunto de vetores de treinamento não é suficiente para representar o problema analisado ou ainda, o processo apresentar não-linearidades que não ficaram contempladas nos vetores de treinamento. Neste caso, sugere-se que seja analisado melhor o comportamento do processo em malha aberta e desta forma, considerando-se as inflexões observadas na relação entrada/saída do processo, optar pela simples introdução de novos vetores de treinamento, ou então, subdividir o problema, especializando uma rede para cada trecho crítico do processo.

De qualquer forma, observando-se uma deteriorização dos resultados dos testes, a primeira providência a ser tomada é a redução do tamanho da rede, ou seja, deve-se iniciar o treinamento usando-se o menor número possível de neurônios na camada intermediária e ir acrescentando neurônios a medida que é necessário. É importante ter-se em mente que um processo de treinamento quase sempre se inicia com pesos obtidos de uma forma randomizada. Nesta condição, uma única sessão de treinamento não é por si só suficiente para que se tenha uma perfeita avaliação sobre questões como sobretreinamento ou sobredimensionamento. Sempre deve-se tratar a questão do treinamento sem negligência, ou seja, repetir o treinamento quantas vezes for possível de ser realizado até que se perceba que o resultado final dos testes tenha sido relativamente semelhante entre os vários treinamentos.

Em relação ao Ambiente proposto, observa-se que a conjunção do algoritmo genético na fase preliminar do treinamento associado ao ajuste automático da taxa de aprendizagem do algoritmo "backpropagation", levaram a quase que a totalidade dos testes efetuados sempre a uma condição evolutiva coerente, praticamente dispensando retreinamentos. Entretanto, quando se buscou treinamentos sucessivos, percebeu-se uma certa repetibilidade dos resultados. O mesmo se aplicando em relação ao treinamento, acontecendo principalmente quando se opta pelo ajuste automático da taxa de aprendizado, onde o treinamento somente será encerrado quando efetivamente for atingido o mínimo da região da função do erro.

Na figura 10 é apresentado um fluxograma que identifica a seqüência mais efetiva para se realizar o treinamento de uma rede neural genérica.

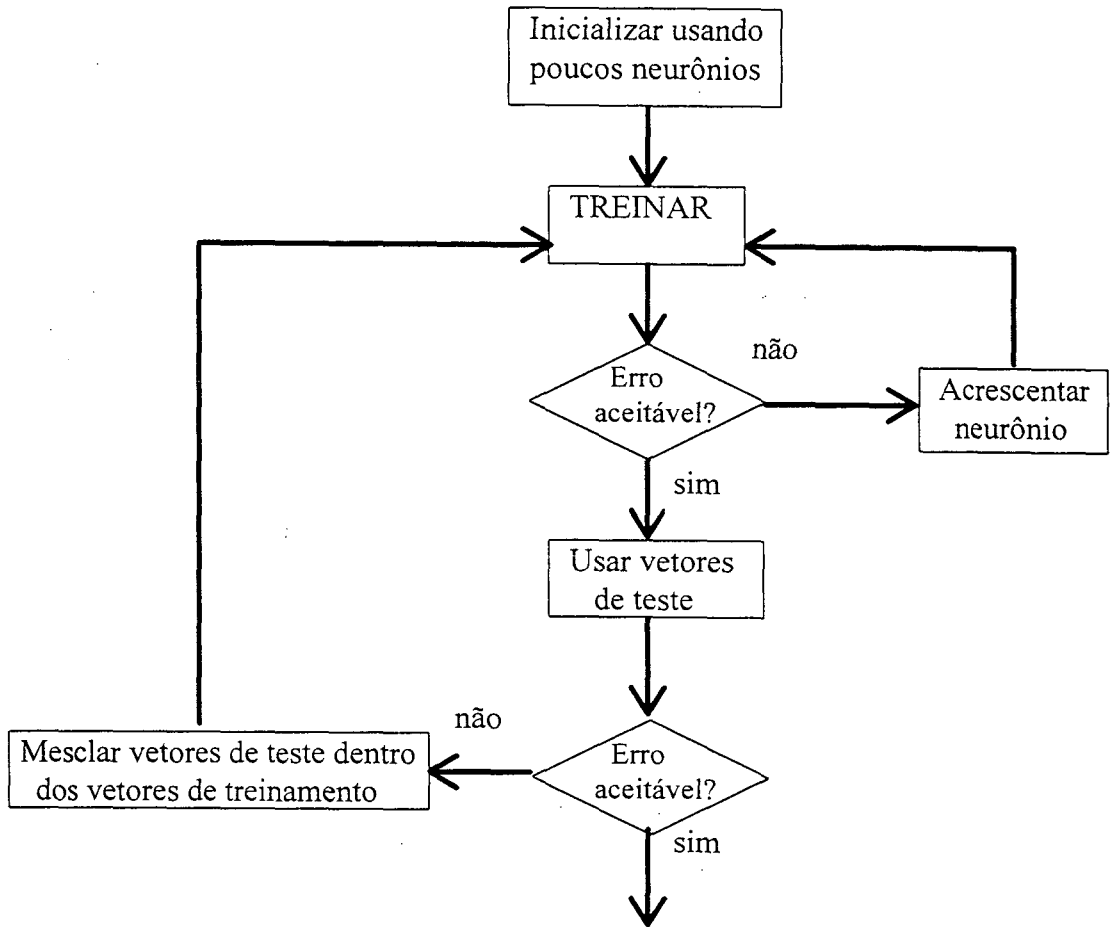


Figura 10 - Seqüência de treinamento

Disto, pode-se concluir que é fundamentalmente importante a determinação de um conjunto adequado de vetores de treinamento. Pode-se, em razão de um conjunto de vetores de treinamento não representativo, provocar uma situação de sobredimensionamento da rede e ainda criar uma falsa idéia de sobretreinamento. Normalmente o sobredimensionamento ocorre quando o tamanho do conjunto de treinamento é pequeno em relação ao número de neurônios das camadas intermediárias.

5.2 Interpretando os Pesos

As redes neurais são freqüentemente criticadas pois é muito complexo entender como exatamente uma rede funciona simplesmente olhando para os seus pesos.

Pode-se dizer que neurônios com alto valor absoluto de entrada devem ser importantes, ao passo que, neurônios com valor absoluto de entrada próximo de zero são por conseqüência, pouco importantes.

Em termos gerais, uma conexão de entrada com valor alto não significa necessariamente que este neurônio é importante. É possível que este neurônio apresente conexão de saída com um valor próximo a zero, ou então, numa camada seguinte, apresente alguma conexão com valor bem próximo a zero, o que tornaria este neurônio não importante.

Similarmente, pesos próximo de zero não necessariamente significam que estes possam ser desprezados. Neste caso deve-se considerar a contribuição dos demais pesos. Desde o trabalho proposto por Rumelhart (1988), foram publicados trabalhos [2,3] que fazem considerações a respeito da importância isolada de um neurônio, o que a princípio, ficam no âmbito simplesmente das considerações e não funcionam como regra. Desta forma, critérios de simples poda de neurônios em função dos valores dos pesos associados não são necessariamente eficientes, mesmo que isto possa significar um relativo aumento da velocidade de treinamento.

Geoffrey Hinton [3] foi o idealizador do primeiro método de representação de neurônios que se popularizou e que passou a ser denominado diagrama de Hinton. Hoje existem variações deste método que basicamente contém várias caixas ou quadrados individuais que representam os pesos. Estes quadrados são preenchidos com cores entre o preto e branco, passando pelo vermelho e azul, de acordo com a importância da conexão. Esta representação pode ser feita para a rede como um todo ou por neurônio.

Um método simples porém eficiente de análise de importância de uma conexão é a denominada análise de sensibilidade. Neste método é fixado o valor de uma conexão e analisado o erro final após o treinamento. Se o erro permanece praticamente o mesmo que com a conexão livre, pode-se concluir que esta conexão não é importante. A dificuldade de verificação deste critério reside no fato que esta avaliação deve ser obrigatoriamente feita para um grande conjunto de padrões de treinamento, visto que, pode ocorrer casos em que para um determinado padrão o erro diminua e para outro e até mesmo para a maioria, este erro aumente significativamente.

Também não é fácil decidir qual das conexões devem ser fixadas. Normalmente deve-se evitar as conexões com valores próximos a zero ou com valores bem altos, ou seja, aquelas conexões que estão com um nível de ativação muito fraco ou estão totalmente ativadas. Isto pode levar entradas à saturação fazendo com que outras entradas acabem por não serem consideradas. Existem variações deste método, entre elas, fixação de uma conexão em valores próximo ao extremo inferior e superior e, da mesma forma, analisar o erro final. Ainda, é possível fixar todas as outras conexões e manter somente uma livre.

O autor considera importante fazer uma investigação sobre os valores dos pesos associados a cada neurônio tão logo encerrada uma etapa de treinamento. Esta avaliação pode ser feita simplesmente observando a proporcionalidade dos valores das conexões. Isto permite que se tenha uma sensibilidade maior com relação a importância e influência das entradas externas da rede, e com um pouco de experiência, poder inferir se a rede está ou não super dimensionada.

5.3 Projetando-se o Conjunto de Treinamento

Sem dúvida, este é um dos pontos mais fundamentais na implementação de uma rede neural, ou seja, a escolha adequada dos vetores de treinamento, tanto em quantidade como em qualidade.

O primeiro aspecto que deve ser considerado num conjunto de vetores de treinamento é o tamanho, ou seja, quantos vetores são necessários para representar o problema a ser analisado, visto que tem influência direta no tempo de treinamento. Por um lado, não faz muito sentido, gerar um número de vetores maior do que a rede pode classificar.

Basicamente, o conjunto de treinamento deve satisfazer a dois pontos principais:

1. Todas as classes devem estar representadas;
2. Dentro de cada classe, cada variação deve estar adequadamente representada.

Uma rede neural é bastante sensível ao problema do sobretreinamento devido principalmente a seu grande número de parâmetros. Por exemplo, numa rede com 25 neurônios de entrada e 10 neurônios na camada intermediária, considerando-se ainda o neurônio de "threshold", tem-se cerca de 260 parâmetros livres, não considerando ainda os pesos da camada de saída. Em [3] é sugerido que o conjunto de treinamento deva ter para este caso, 500 exemplos ou mais, considerando que para cada duas classes deve-se ter um neurônio. Isto implica dizer que, quanto maior a rede utilizada, tanto maior deve ser o conjunto de treinamento. Para um conjunto de treinamento muito grande, corre-se o risco de ter padrões de treinamento repetidos.

Uma maneira prática de determinar uma quantidade adequada de número de vetores de treinamento, é iniciar utilizando um número pequeno de vetores, de modo que a rede possa simplesmente responder aos casos mais fáceis. A partir daí se introduz mais vetores de treinamento buscando reforçar aqueles pontos onde a rede não representou a contento. Isto evita que se gere vetores tidos como supérfluos. Infelizmente, isto só pode ser definido durante os testes, e a partir dos resultados obtidos, redimensionar o conjunto de vetores. Para que este trabalho seja o mais satisfatório possível, deve ser evitada qualquer influência devida ao fator humano. O que se quer dizer com isto, que em situações onde os testes são realizados a partir de vetores escolhidos por um único especialista, e esta escolha é feita similarmente a maneira utilizada na identificação dos vetores que forão utilizados para o treinamento, pode ser que imbutido nestes vetores, haja uma tendência de captura de uma determinada característica do problema que pode ter sido previlegiada pelo especialista. Nestes casos, sugere-se que os vetores de treinamento e teste sejam obtidos por diferentes especialistas.

No Ambiente proposto são oferecidos recursos adicionais de modo a facilitar a forma de criação dos vetores de treinamento. Quanto aos testes, estes são realizados diretamente sobre o processo, o que evita sobremaneira qualquer vício humano gerado no processo de captura destes vetores em geral.

O autor considera que inicialmente deve-se procurar obter a relação entrada/saída do processo. Isto permite que seja identificado problemas de não linearidade dentro da faixa de controle. Esta situação é de grande valia principalmente como informação para que se possa subdividir o problema do

controle não em uma única rede, e sim em várias redes, cada uma especializada numa faixa do controle. Em suma, identificado as não-linearidades, pode-se especializar uma rede para cada sentido de variação da derivada da função observada, ou melhor, subdivide-se o problema, especializando uma rede para os trechos onde a derivada é crescente, outra rede para onde a derivada é decrescente e por fim uma outra rede ainda para aqueles trechos onde a derivada é praticamente constante. Isto evita que se tenha que construir uma rede complexa para atender todas as inflexões ou não-linearidades que o processo porventura contenha.

Neste procedimento, a quantidade de vetores a serem capturados se resume a praticamente três ou quatro vetores por rede utilizada, ou seja, um vetor para cada extremo superior e inferior e mais alguns vetores intermediários.

Além do exposto nos parágrafos anteriores, outra consideração importante deve ser feita levando-se em conta a característica do processo analisado. Normalmente, numa condição de variação da excitação, quando se eleva a entrada do processo não é exatamente a mesma de quando se reduz o sinal aplicado a esta entrada, isto para um mesmo valor de saída, criando um efeito tipo histerese. De uma outra forma, o valor correspondente a saída do processo depende não somente do valor atual aplicado a entrada deste processo, mais também ao valor anterior no tempo desta entrada. Exemplificando, supondo um ponto de entrada genericamente identificada assume um valor 50, a saída do processo corresponde a um determinado valor que depende em parte do valor do instante anterior. O valor da saída deste processo é diferente, se no instante anterior mencionado, a entrada tivesse valor igual a 60 em vez de 40.

Neste caso, é importante que nos vetores de treinamento estejam contemplados tanto a excitação crescente como a decrescente. Observou-se, em função dos vários testes realizados, que uma relação de três para um, ou seja, três vetores de excitação crescente para um de excitação decrescente, já atende o propósito para o caso do processo analisado no capítulo 6, isto para cada uma das redes especializadas. Logo, deve-se capturar vetores de treinamento para variações crescente e decrescente da referência do controlador.

Ainda com relação a captura dos vetores de treinamento, ocorre uma certa tendência de que os últimos vetores inseridos tenham uma importância

relativamente maior que os primeiros. Isto se deve ao fato de que durante o algoritmo de "backpropagation", o erro é ajustado em passos pequenos para cada vetor do conjunto de vetores de treinamento. Como o encerramento de um processo de treinamento depende da análise final do erro, e esta só ocorre quando é atingido o último vetor da tabela de vetores de treinamento, existe então uma pequena tendência destes vetores serem relativamente mais dominantes que os primeiros.

5.4 Implementação

5.4.1 Algoritmos tipo Gradiente Descendente

Este grupo, se refere ao algoritmo "backpropagation" e o "simulated annealing".

Quanto ao algoritmo "backpropagation", no que tange a sua implementação, seguiu-se a idéia básica deste algoritmo, somente acrescentando alguns recursos que tem como objetivo auxiliar o usuário do Ambiente durante o treinamento. Neste ínterim, destaca-se o ajuste automático da taxa de aprendizagem, que tem o seu valor reduzido a cada instante que ocorre uma não-convergência da função objetivo. Esta redução é feita a partir de uma constante definida pelo usuário.

Como critério, e por avaliação empírica, foi adotado que os pesos iniciais randomizados não tem valor em módulo maior que dez.

Em paralelo com o cálculo do erro de cada iteração é sinalizado se está havendo convergência ou não dos valores do erro global. A não convergência é indicada através de um sinalizador específico (flag). A cada situação em que ocorre uma sinalização de não-convergência, é dividido o valor atual da taxa de aprendizagem por um fator, fator este predefinido. Na prática, decréscimos que provocam alterações entre cinco e dez porcento no valor da taxa de aprendizagem a cada situação de não-convergência demostraram bons resultados. Observar que nesta situação a taxa de aprendizado pode assumir um

valor tão pequeno quanto se queira, e até atingir o zero numérico computacional se assim se desejar. Durante as inúmeras operações de treinamento realizadas, valores da taxa inferiores a 0,000001 (um micronésimo) mostram de pouco interesse prático, pois já não se verificam mais variações significantes nos pesos da rede.

Na escolha do valor adequado da taxa de aprendizado para o algoritmo "backpropagation", o autor sugere iniciar com uma taxa relativamente alta (>1), deixando ao próprio Ambiente a escolha do valor adequado, permitindo com que a convergência ocorra de uma maneira relativamente rápida sem nenhum prejuízo com relação à solução final obtida.

Já com relação a taxa de momentum, nenhum ajuste automático está previsto. O que se observa é que esta taxa deve ser mantida num valor intermediário dentro de sua faixa, ou seja, $0 \leq \alpha \leq 1$. Para valores muito pequenos da taxa de aprendizagem podemos desprezar a taxa de momentum. Em [25] é sugerido um método que relaciona as duas taxas.

No que diz respeito ao "simulated annealing", este treinamento funciona conjugado ao "backpropagation", sendo então denominado de randomização controlada, e é apresentado como uma opção de treinamento durante a utilização do algoritmo "backpropagation" também conhecido como Regra Delta.

Foram incluídas outras opções de randomização total ou não controlada, bem como recursos para salvar pesos durante o treinamento, ou então, utilizar pesos de uma outra rede já parcialmente treinada.

Durante o treinamento utilizando o algoritmo "backpropagation", a saída pode permanecer imobilizada devido a algum padrão de treinamento não coerente. Isto pode ocorrer devido a propriedades da função sigmóide, que atinge uma quase saturação para valores próximos de zero e de um. O observado é que durante algumas iterações não ocorre alteração significativa na saída do neurônio, prejudicando em parte o processo de treinamento. Este efeito ocorre quando o valor da derivada dado por $o_j(1-o_j)$ passa a ser muito pequeno, onde o_j corresponde a função de ativação.

A forma empregada para contornar o problema apresentado da derivada para valores baixos, é de provocar uma compressão linear dos valores situados entre 0 e 0,1, fazendo uma transformação deste espaço para valores entre 0,05 e 0,1. Caso o valor da derivada seja inferior a 0,1, passa-se a usar a equação abaixo:

$$nova\_derivada = 0{,}05 + 0{,}5 \times valor\_derivada \tag{43}$$

Desta forma garantimos que não teremos derivadas com valores muito baixo, além de 0,05. Em [16] é apresentado um método onde é adicionado um fator constante positivo (offset) à expressão $o_j(1-o_j)$. O fator sugerido no artigo é 0,1.

Visto que, do ponto de vista computacional a operação realizada na equação (43), é extremamente rápida frente a outros passos do próprio algoritmo, foi mantida a solução aqui sugerida por apresentar pouca perturbação ao comportamento da rede durante o treinamento, não apresentando degraus de transição de valores quando da introdução do valor constante diferente de zero como no caso apresentado em [16].

Neste trabalho é assumido que todos os neurônios têm a mesma função de ativação, tanto na camada de saída como nas camadas intermediárias. Foram ensaiadas funções de ativação distintas entre camadas, e a princípio, não foi obtido alteração significativa no comportamento final da rede que justificasse que estas alterações devessem ser mantidas. No caso, conservou-se unicamente a função de ativação logística. Normalmente este é o procedimento adotado. No entanto, deve-se enfatizar que é possível a utilização de outras funções de ativação sempre que necessário. Algumas aplicações utilizam a função identidade, $f(x) = x$, como função de ativação dos neurônios da camada de saída. O efeito resultante desta ação é concernente a uma redução a imunidade a ruído. Em algumas situações podem ser utilizadas funções de compressão para atenuar este efeito. Naturalmente uma rede não tem capacidade de aprendizado caso seus neurônios sejam lineares e o conjunto de treinamento não seja linearmente separável [3]. O benefício de se utilizar o neurônio da última camada linear acontece em aplicações do tipo filtros autoassociativos, quando se utiliza técnicas de regressão, entre outras, que foge a expectativa deste trabalho.

5.4.2 Algoritmo Genético

Descreve-se a seguir a implementação deste algoritmo.

Inicialmente uma população pobre de indivíduos é criada (aqui denominada de cromossomos). São gerados "n" conjuntos de pesos designando cada conjunto como um cromossomo, correspondendo ao período de *inicialização*. O conjunto de vetores utilizado para o treinamento é então aplicado à rede utilizado-se cada um dos conjuntos de pesos. Para cada conjunto de pesos (ou cromossomo) é armazenado o erro médio quadrático resultante.

A primeira ação sobre estes pesos é a denominada *elitização* ou seleção dos pais, onde se elimina os 25 % piores conjuntos de pesos e duplica-se os 25 % melhores. Desta forma o total de vetores permanece o mesmo, pois os 50 % restantes são mantidos inalterados.

O método da elitização contempla de uma certa maneira a técnica denominada "*overinitialization*". Se a população inicial é randomicamente gerada, certamente há indivíduos com características gerais mais pobres que outros, ou seja, o material genético de alguns destes indivíduos, com certeza, tem uma aptidão ou adequabilidade extremamente baixa, e é quase garantido que estes indivíduos têm pouca contribuição para as futuras gerações. Uma forma de se melhorar a primeira geração, é criar randomicamente uma população maior do que a requerida, e então, descartar os piores indivíduos.

Por exemplo, supondo que o tamanho da população que se deseja é de 100 indivíduos, são gerados inicialmente 150 indivíduos e em seguida são eliminados os 50 piores, mantendo-se então a população alvo de 100. Desta forma, o processo de seleção que ocorre a partir daí, é inicializado com grupo genético de melhor qualidade. Tipicamente o número de iterações necessárias até que se complete um ciclo de treinamento genético, quase sempre extrapola a 100 iterações, ou seja, após este número de iterações muito pouco irá restar de algum indivíduo supostamente doente. Sendo assim, no Ambiente sempre é mantido um número fixo de cromossomos. Outra razão e talvez a principal de se manter fixo o número de cromossomos foi devido a restrições impostas pelo próprio ambiente de desenvolvimento de software no que tange o problema de espaço para locação de dados.

O algoritmo genético é indicado para a fase preliminar de treinamento, ou seja, quando se está realizando os primeiros ciclos de otimização da rede neural. Nesta situação muito pouco efetivamente se ganha em termos de tempo de treinamento relativo ao tempo total do treinamento. É significativamente mais importante que o algoritmo que se encarrega de fazer o ajuste mais especializado ou fino dos pesos, neste caso, o "backpropagation", que este seja o mais eficiente, pois este método é basicamente responsável pelo treinamento on-line.

Neste trabalho foi suprimida a fase denominada "crossover" ou recombinação, visto que ela não amplia de forma significativa o espaço de busca, o que representaria de qualquer maneira um custo computacional. Esta etapa corresponderia a troca de valores de pesos de posições distintas de um mesmo vetor ou entre vetores. A escolha das posições é feita de maneira aleatória. Não existe uma regra que defina o número de trocas a serem executadas. Também é sugerida a operação de reprodução, ou seja, gerar (normalmente dois) filhos a partir das características dos pais (normalmente dois também). São entrelaçadas as características dos dois pais formando os dois filhos. Cada filho passa a ser formado com a metade dos genes de um pai e a metade do outro. As metades excedentes dos dois pais formam o outro filho, não havendo perda do material genético. Este método é denominado de "crossover" de um ponto. Também é sugerida [3] a divisão em quatro blocos de cada pai. Sendo assim um filho seria resultante do bloco um e três de um dos pais e do bloco dois e quatro do outro pai. Os blocos restantes dos pais formarão o outro filho. Este método é denominado de "crossover" de dois pontos. Em algumas soluções é feita uma mescla entre o método de um ponto e o de dois pontos afetando a probabilidade de um par de genes permanecerem juntos.

A fase seguinte é denominada de *mutação* e consiste em substituir também de forma aleatória, dentro da matriz de pesos (conjunto de cromossomos ou então população), alguns de seus valores (gene), isto é, está se introduzindo uma nova informação genética a cada um dos indivíduos. A taxa de mutação ou a quantidade de novas informações genéticas a serem introduzidas na população é um parâmetro arbitrário. O processo injeta informação nova numa população heterogênea, e é bastante conveniente, pois não existe garantia que a solução do problema esteja no universo de pesos vigente. Por ser aleatória, a mutação pode

também destruir um bom vetor de pesos antes que este possa ser duplicado. Na prática, verifica-se que uma taxa elevada de mutação provoca oscilação nos valores do erro. O conceito de taxa alta ou baixa pode ser medido proporcionalmente ao número de cromossomos escolhidos. Deve-se buscar uma taxa preferencialmente um pouco inferior ao número de cromossomos, principalmente caso não se esteja utilizando nenhum método que evite o problema da não convergência para valores pequenos do erro.

Encerrada a fase de mutação é dito terminada uma geração. A partir deste ponto, se reinicia o processo de elitização/mutação até que seja alcançado o erro esperado ou seja atingido o número de iterações predefinido.

No Ambiente proposto, a alteração dos pesos a partir das mutações pode ser feita de uma *forma controlada*, ou de *forma elitizada*, como explicaremos a seguir. Isto permite contornar de maneira efetiva a maior dificuldade que este algoritmo apresenta que é o problema associado a divergência do erro quando este atinge valores relativamente pequenos. Na *forma controlada* ou combinada, a alteração no processo de mutação só se realiza se o novo peso apresentar um erro médio quadrático inferior ao anteriormente existente. Na *forma elitizada*, a alteração dos pesos só acontece sobre os cromossomos piores, ou seja, são preservados intactos os 25% melhores cromossomos. Em ambas as situações é facilitado o trabalho do usuário, visto estabelecer um controle sobre a não convergência dos pesos de forma automática, não exigindo a supervisão contínua do usuário. Ambas as formas apresentam desempenho dentro do esperado, somente que a primeira exige mais esforço computacional que a segunda. Na forma controlada é facultada a introdução de informação diferente da esperada, ou seja, esporadicamente, a uma probabilidade de 0,1, é relaxado o critério de substituição dos pesos, permitindo que ocorra sobre algum dos cromossomos a introdução de uma informação nova, livre de controle. Como resultado imediato, pode esta operação apresentar um aparente retrocesso no treinamento. De uma forma geral, esta operação permite que não ocorra um estreitamento abrupto do espaço de busca, o que poderia ocasionar o problema da convergência prematura.

5.5 Pré-treinamento

5.5.1 Introdução

Um problema muitas vezes de difícil solução que se enfrenta na realização de um controlador para controlar um processo real qualquer, é como preparar o controlador antes de conectá-lo ao processo propriamente dito, para que se possa efetivamente fazer com que este controlador aprenda a dinâmica do processo (observar que a rede inicia com pesos randomizados, cuja relação entrada/saída é totalmente desconhecida).

5.5.2 Desenvolvimento

A solução adotada foi a de gerar inicialmente vetores de teste que são obtidos ensaiando-se o processo em malha aberta, imaginando que o controlador estaria numa plena condição de repouso ou "steady-state". Conhecendo-se os valores dos sinais de entrada do processo e a corresponde relação com a saída do processo, pode-se criar um conjunto de vetores, denominados de "true vectors" (pois representam relações **estáticas** de E/S sempre válidas) que são utilizados na fase de pré-treinamento da rede controladora. Os valores da tabela são obtidos após o sistema atingir o estado estável (figura 12).

Na figura 11, BF ("Base Factor") é um fator de adaptação do sinal de referência ao sinal de realimentação. Este fator é melhor apresentado no anexo B3, figura 30.

A cada vetor (linha) da tabela acima são acrescidos opcionalmente mais 1 ou 2 vetores, que indicam quanto varia $u_{(k)}$ para uma variação de ± 10 % do valor de $y_{(k)}$. Pode-se definir se a variação de $u_{(k)}$ será direta ou inversamente proporcional a variação de $y_{(k)}$ (efeito de regulação direta ou reversa). Na figura 13 temos a representação da captura de um único "true vector" sendo acrescidos mais dois vetores com efeito de regulação reversa, onde V é um fator de velocidade de ajuste do controlador, ou seja, define o ganho do controlador.

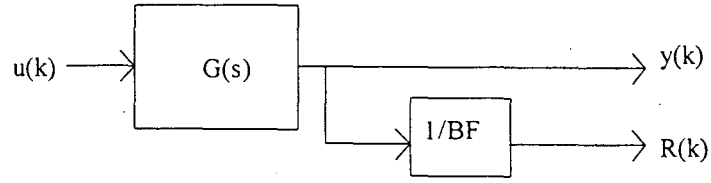


Figura 11 - Configuração para aquisição dos "true vectors"

Estes "true vectors" também podem ser utilizados na fase de treinamento adaptativa funcionando como uma espécie de âncora ou segurança da memória. O que se procura é evitar que, por uma escolha não adequada do método de obtenção automática de novos vetores, venha com isto o controlador a comprometer o processo controlado, devido a introdução de algum vetor ruim dentro do conjunto de treinamento. Assim, a presença dos "true vectors" garantem que não ocorra um treinamento completamente errôneo, mantendo-se em parte sempre as características de controlabilidade predefinidas.

R(k)	y(k)	y(k-1)	u(k)	u(k-1)
r1	y1	y1	u1	u1
r2	y2	y2	u2	u2
.	.	.	.	.
r_n	yn	yn	un	un

Figura 12 - Tabela de "true vectors"

Outra forma de se contornar este problema de perda de memória, é utilizando-se uma opção que controla a variação de cada peso da rede, visto que toda a informação que a rede possui está justamente armazenada nestes pesos.

R(k)	y(k)	y(k-1)	u(k)	u(k-1)
r_1	y_1*1,1	1,1*y_1	u_1/V	u_1/V
r_1	**y_1**	**y_1**	**u_1**	**u_1**
r_1	y_1*0,9	y_1*0,9	u_1.V	u_1.V
.	.	.	.	.

Figura 13 - Acréscimo off-line de vetores de treinamento

É importante salientar que, devido a forma com que é feita a captura destes "true vectors", todas as imperfeições do processo serão automaticamente consideradas, como por exemplo, um transdutor com problemas de precisão, a

não-linearidade dos atuadores e as não-linearidades do processo propriamente dito.

Caso não seja possível realizar o ensaio em malha aberta, os vetores de teste terão que ser editados diretamente pelo especialista. De qualquer forma, uma vez realizado o controlador preliminar, o desempenho deste pode ser corrigido a posteriori pela aquisição on-line de novos vetores de treinamento.

Os vetores assim obtidos são usados para o treinamento inicial da rede neural controladora.

5.6 Aquisição On-Line

O ajuste inicial é refinado posteriormente através da uma aquisição ON-LINE de novos vetores de treinamento, que são acrescentados na tabela de "true vectors". Os valores originais não são removidos e a tabela é apenas ampliada (figura 15).

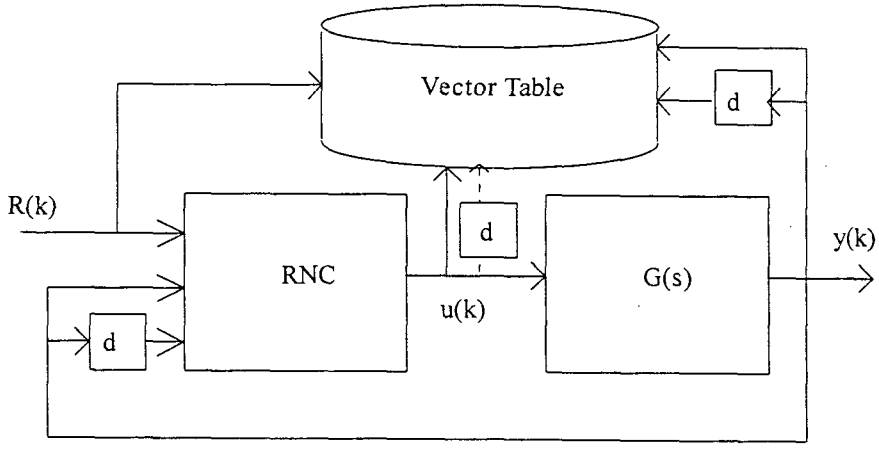


Figura 14 - Aquisição on-line de novos vetores de treinamento

É importante salientar que o Ambiente proposto possue as ferramentas necessárias para a obtenção dos vetores de treinamento de forma on-line (figura 14). Isto permite que todas as características dinâmicas do processo sejam automaticamente consideradas e também se torna uma ferramenta indispensável na realização de controladores com estrutura adaptativa.

R(k)	y(k)	y(k-1)	u(k)	u(k-1)
r_1	y_1	y_1	u_1	u_1
r_2	y_2	y_1	u_2	u_1
r_3	y_3	y_2	u_3	u_2
.	.	.	.	.
r_n	y_n	y_{n-1}	u_n	u_{n-1}

Figura 15 - Tabela ampliada com vetores adquiridos on-line

Dentro do contexto de controladores adaptativos, devemos a cada instante ler novos vetores e utilizá-los no retreinamento da rede. Ocorre que não podemos ampliar o número de vetores de teste de uma forma ilimitada, naturalmente por várias razões e uma bastante significativa que é o tempo de treinamento. Seguindo este raciocínio, tão pouco podemos detectar qual o vetor que deve ser alterado, pois pertencia a um conjunto de valores anteriores ao momento em que foi disparado o processo de captura automática. A solução de trocar todos os vetores também não deve ser utilizada indiscriminadamente pois seria como se a rede perdesse a memória anterior.

Em face a esta problemática, criou-se um conjunto de opções que, se corretamente combinadas, podem contornar a totalidade destes problemas. Estas opções estão melhor descritas no anexo B.

O Ambiente proposto permite fazer um controle sobre o valor instantâneo da referência no sentido de facilitar a obtenção do vetores de forma on-line. A referência do controlador neural pode variar de uma forma manual, com definição da condição de pré- e pós-disparo, ou automaticamente seguindo uma variação senoidal, com limites predefiníveis e número de repetições controladas. A referência também pode seguir um conjunto de valores predefinidos num vetor de valores com repetição controlada. Estas repetições permitem determinam quanto tempo será mantido um mesmo valor de referência antes de mudar automaticamente para o próximo, estabelecendo patamares de valores que é muito significativo devido aos retardos associados à própria dinâmica do processo.

A alteração dos vetores pode ainda ser feita na forma de blocos ou total, permitindo estabelecer um critério de suavidade de adaptação. Dividir o

processo de captura em blocos auxilia quando se deseja fazer a captura das características dinâmicas, permitindo que seja incorporada esta dinâmica pela aplicação de vários sinais, por exemplo, degrau crescente, degrau decrescente, variação senoidal, e todos estes vetores adquiridos ocuparão o mesmo conjunto de treinamento separados pelos respectivos blocos.

É importante lembrar que os vetores de treinamento "true vectors" sempre serão mantidos inalterados. Pode-se, através de um editor de texto, copiar algum vetor capturado dinamicamente e inserí-lo junto com os "true vectors". Nesta situação, este vetor que foi inserido também será mantido no conjunto de vetores de treinamento, não sendo mais alterado durante novas capturas dinâmicas.

Uma situação desinteressante é possuir vetores iguais ou muito semelhantes. Durante o processo de aquisição de novos vetores de teste, podemos filtrar vetores que se assemelhem aos já existentes de uma maneira controlada, podendo ser ajustado o critério de semelhança.

Suponha que o controlador em questão permaneça por muito tempo em uma única posição de controle. Num processo de captura automática, típica de controladores adaptativos, o que poderá resultar desta situação é que o controlador irá aprender muito este ponto ou região e acabar aprendendo menos as demais, ocorrendo o risco de haver um desaprendizado das outras regiões principalmente, após sucessivas aquisições dos vetores de treinamento.

Para solucionar este problema foi criada uma opção que só habilita a substituição de um vetor de treinamento se o valor atual da referência estiver na vizinhança do valor da referência de algum vetor de treinamento. Desta maneira é possível manter uma correlação da distribuição dos vetores de treinamento de acordo como foram definidos ainda na fase preliminar.

Outra opção implementada diz respeito a condição para se iniciar um novo processo de aquisição de novos vetores e por sua vez disparar um novo ciclo de treinamento. Esta cadeia pode ser acionada de uma maneira automática ou através de um disparo externo ("external trigger"). Este disparo se refere a uma determinada variação de uma das entradas previamente definidas. Caso o valor

desta entrada ultrapasse a um determinado percentual do valor atual, será então desencadeado o processo de aquisição de forma automática.

Quando o problema tratado apresenta muitas não linearidades, ou seja, é de alta complexidade, obrigará que se gere uma rede neural proporcionalmente complexa para executar o controle de forma efetiva. No Ambiente proposto, é possível subdividir o problema principal em problemas menores, de forma a especializar redes em determinadas faixas dentro da região de atuação do controlador. Isto se aplica também para os "true vectors". Naturalmente cada uma destas redes será proporcionalmente mais simples, e portanto, de resposta mais rápida. O custo de chaveamento entre uma rede e outra é praticamente desprezível em relação ao tempo de resposta da própria rede, ou seja, fica praticamente transparente este chaveamento para o restante do sistema. É possível realizar sobreposição entre redes de modo que a transição de uma para outra seja mais suave.

Uma forma indireta de se reproduzir sobre as redes neurais um comportamento similar ao de um derivador quando se trabalha com controladores PIDs é o resultante da opção denominada como "*Delay accelerator* ". Este fator é usado para multiplicar os valores do instante (k-1), (k-2), ..., ou seja, dos instantes de amostragem anteriores ao atual. Este efeito aparecerá a segunda potência para o instante (k-2), terceira potência para o instante (k-3) e assim sucessivamente. Observe que o "*delay accelerator*" só opera sobre os atrasos ou "delays".

Outras opções como variação da velocidade de resposta da rede após esta já ter sido treinada foram implementadas no sentido de facilitar a visualização dos dados adquiridos e se estabelecer um efetivo controle sobre os parâmetros de treinamento e de aquisição. Estas opções estão integralmente detalhadas no anexo B.

Devemos ressaltar que os "true vectors" são obtidos também de uma forma on-line. O que diferencia os "true vectors" da captura dita on-line, é que no caso específico dos "true vectors" a obtenção destes é feita em malha aberta, pressupondo-se que o processo esteja perfeitamente estável no instante da captura. Já na dita captura on-line, os vetores são adquiridos em malha fechada. Isto permite que pela variação do sinal de controle aplicado ao controlador/planta seja possível adquirir vetores de treinamento que possuam

incorporados a característica dinâmica da planta, representada aqui pela diferença entre os valores de $y(t)$ para cada instante de amostragem.

5.7 Realização do Controlador

5.7.1 Introdução

Na literatura especializada encontra-se formas alternativas de se realizar controladores neurais, alguns até utilizando um modelo matemático de referência do processo. Neste caso, é usado o poder de adaptação das redes neurais e é mantida a necessidade de se conhecer o modelo matemático em questão com um certo grau de precisão sob pena de comprometer os resultados esperados.

A linha adotada neste trabalho foi a de se realizar um controlador puramente neural, sem a necessidade do conhecimento prévio do modelo matemático envolvido, bastando tão somente ter-se uma relativa idéia da ordem deste modelo, que também não é indispensável. O conhecimento da ordem do modelo simplesmente irá diminuir a quantidade de treinamentos/ensaios até se obter uma solução satisfatória.

Foram utilizados dois modelos de controlador neural, o denominado controle direto e o controle indireto, que apresentaremos a seguir.

5.7.2 Controle Indireto

Neste modelo são usadas duas redes neurais, uma denominada de rede identificadora que representa o comportamento do processo, rede esta que é utilizada no treinamento de uma segunda rede denominada rede controladora, que por sua vez é o controlador propriamente dito.

A razão da existência da rede identificadora é para retropropagar o erro global do processo (referência - saída da planta) de modo que este erro possa ser sentido na saída da rede controladora durante o treinamento.

O Ambiente prevê as condições necessárias para a implementação das duas redes.

Inicialmente é treinada a rede neural identificadora (RNI) usando o próprio Ambiente para se fazer a aquisição dos vetores de treinamento necessários para este caso, de acordo com a figura 16. Este treinamento é feito off-line, disparado após a aquisição de vetores de treinamento (figura 17):

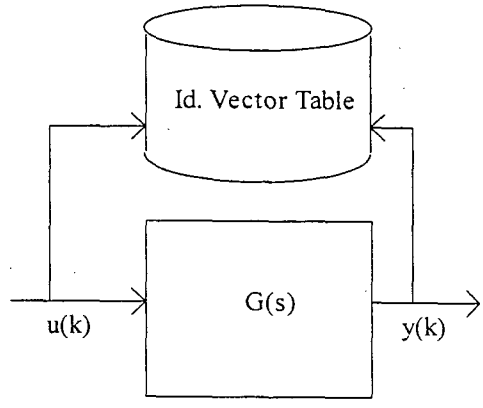


Figura 16 - Aquisição de vetores de treinamento para a Rede Neural Identificadora (RNI)

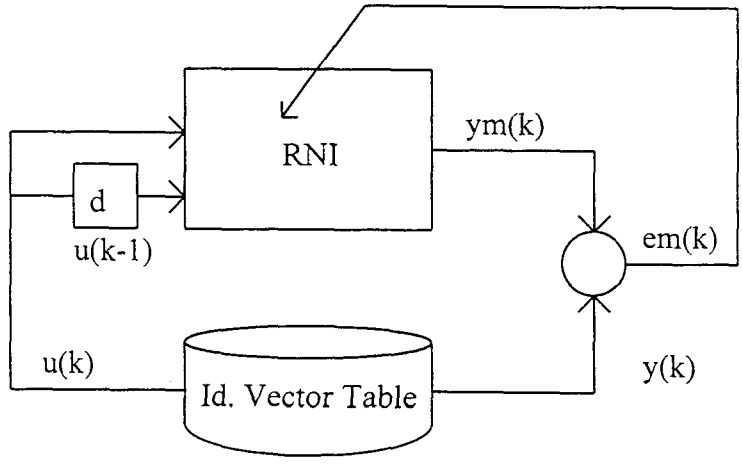


Figura 17 - Treinamento da RNI

O treinamento é concluído quando $e_{m(k)}$ (erro de modelo) é menor que um valor especificado.

Encerrado o treinamento, o comportamento da rede identificadora pode ser comparado em tempo real ao comportamento do processo a ser controlado o que permite verificar de uma maneira bastante explícita o resultado do treinamento.

Depois do treinamento da rede neural identificadora (RNI), o esquema da figura 18 é montado, de forma a treinar um segundo controlador neural (RNA). O erro de modelo é retropropagado através da RNI, sem alterar seus pesos, mas ajustando os pesos da RNA. O aprendizado da RNA pode ser feito em paralelo com o controle em tempo real ou não. O que se quer dizer é que é possível manter uma rede controladora efetivamente controlando a planta em tempo real e manter outra rede em paralelo sendo treinada. O treinamento é baseado nos dados contidos na tabela de vetores de treinamento previamente adquirida, e desta forma, não há interferência da rede que está sendo treinada na rede que está efetivamente controlando a planta. O controle em tempo real pode ser desativado, permitindo que o treinamento se processe de uma forma mais rápida.

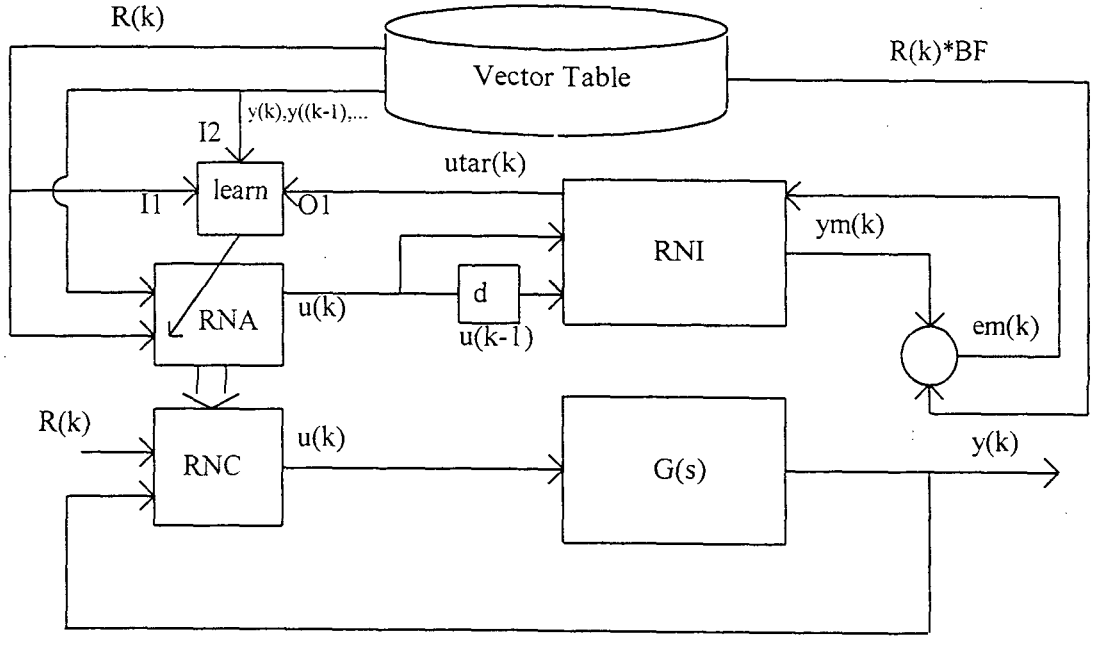


Figura 18 - Esquema para Controle Indireto

No modo adaptativo o Ambiente captura os vetores de treinamento em tempo real gerando então uma tabela que será utilizada durante o treinamento da RNA,

sendo que este treinamento pode ser disparado de forma automática de acordo com um ciclo iterativo de captura/treinamento. Para a captura, pode-se previamente definir a quantidade de vetores a serem capturados, as condições de captura e de atualização sobre os vetores já existentes.

Quando $e_{m(k)}$ fica menor que um valor especificado, os pesos da RNA são copiados na RNC, atualizando o controlador. Nesta configuração a rede neural identificadora somente é utilizada no treinamento para os vetores capturados on-line. Quanto o vetor de treinamento que está sendo aplicado a RNA é um "true vector", o valor alvo de saída [utar(k), utar(k-1), ...] é obtido diretamente da tabela de vetores, não sendo, por conseguinte utilizada a RNI.

Durante a implementação do Ambiente foi criada uma opção de geração de vetores de treinamento para a RNA de forma totalmente off-line. Esta opção foi considerada com eficiência inferior a obtida diretamente com o método já descrito, utilizando-se os "true vectors". O objetivo inicial seria o de fazer um pré-treinamento da rede neural controladora antes de conectá-la diretamente ao processo. Após vários ensaios isto se demostrou desnecessário, já que com somente o treinamento obtido a partir dos "true vectors" se obtém um controlador com desempenho bastante próximo ao definitivo.

Sugere-se que sejam utilizados um número de "true vectors" suficientes para cobrir a faixa dinâmica de referência. Um número razoável destes "true vectors" seria em torno de 20 vetores para o processo piloto analisado (ver capítulo 6). Desta forma, garante-se que ao se iniciar o processo de aquisição dinâmica (on-line), o controlador irá responder aos comandos de variação do sinal de referência sem no entanto comprometer o processo controlado, e também, permite que as variações provocadas no sinal de referência sejam aceitos plenamente pelo controlador.

O que se procura fazer é iniciar o treinamento utilizando somente os "true vectors" e só após a rede ter sido treinada com estes vetores é que se faz a aquisição dinâmica de novos vetores.

5.7.3 Controle Direto

Neste modelo tem-se unicamente o controlador neural comandando diretamente o processo.

A dificuldade de implementação deste método surge na fase de treinamento do controlador neural quando da captura dinâmica. O problema é como saber qual o valor que a saída do controlador deve ter a cada variação dos valores na entrada do controlador. O erro que surge na saída da planta em relação a um valor de referência deve ser compensado pelo controlador neural de forma semelhante ao que acontece quando se tem um PID clássico. O que ocorre é que tem-se um sistema com uma dinâmica própria (retardos) que devem ser respeitados. Observe que no controlador, uma dada condição de entrada só tem uma determinada saída se as condições anteriores também forem as mesmas, em outras palavras, se analisar um determinado instante, um mesmo conjunto de valores de entrada pode ter saídas completamente diferentes dependendo, como foi dito, das condições anteriores ao instante do evento analisado.

Isto implica que o mapeamento entrada/saída não seja uma coisa trivial.

No controle direto não existe uma rede neural identificadora como no caso do controle indireto. Assim, para a aquisição de novos vetores de treinamento on-line, é necessário estimar o valor desejado para $u_{(k)}$ (u_{target}).

Para fugir a necessidade de se conhecer previamente o modelo matemático da planta de forma a se poder extrapolar qual deve ser o valor instantâneo da saída do controlador para uma determinada condição de entrada, optou-se por utilizar um fator que altere o valor atual da saída do controlador no sentido em que o erro entre a saída do processo e o respectivo valor da referência apontarem.

Este fator, denominado de "ganho", pode ser tanto linear como exponencial e é basicamente adicionado ao valor atual da saída da rede, obtendo-se desta forma o valor alvo desejado para o treinamento.

A expressão referente ao ganho linear (GL), basicamente, adiciona ao valor atual da saída um fator que é igual ao valor atual multiplicado pela diferença

instantânea entre a referência e a saída da planta ($e_{(k)}$) multiplicado por um número pré-determinado (GL).

$$u_{target} = u_{(k)} + u_{(k)}.e_{(k)} .GL \tag{44}$$

Já a expressão do ganho exponencial (GE), multiplica ou divide o valor atual da saída do controlador por um fator que é determinado por um número predefinido (GE), elevado a diferença instantânea entre a referência e a saída da planta ($e_{(k)}$).

$$u_{target} = u_{(k)}.GE^{e(k)} \tag{45}$$

Estes ganhos são ajustáveis e afetam a velocidade de treinamento.

De maneira semelhante ao descrito no modelo de controle indireto, as fórmulas acima somente são utilizadas para os vetores adquiridos de forma on-line. Para os "true vectors", é mantido o treinamento somente a partir da tabela de vetores.

O esquema opera como mostrado na figura 19.

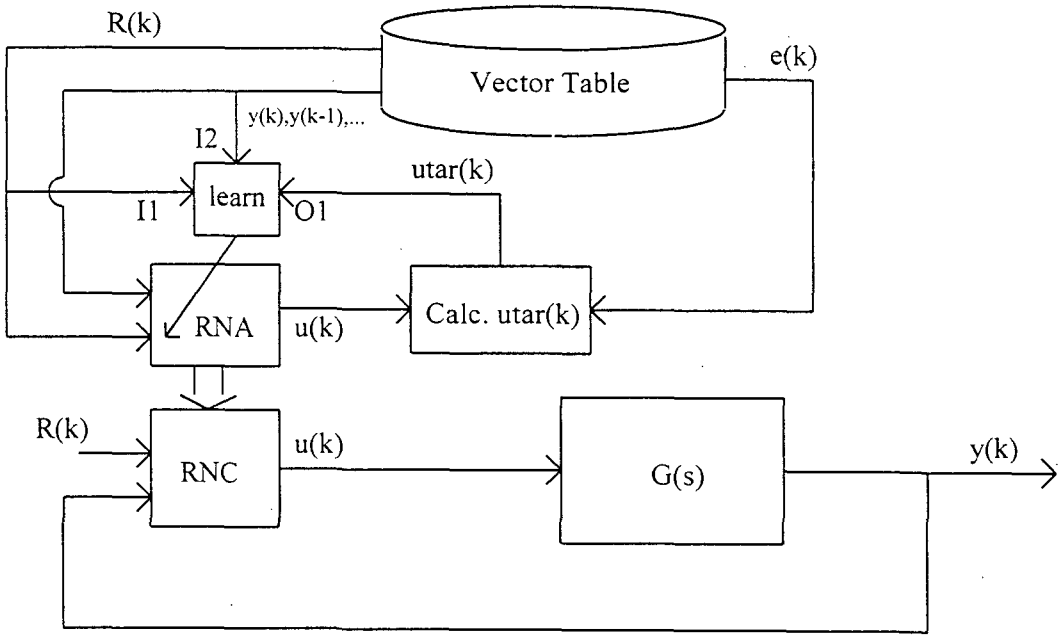


Figura 19 - Esquema para Controle Direto

Os pesos da RNA são copiados para a RNC quando o erro de treinamento fica abaixo de um valor especificado. Para este modelo, e(k) é igual R(k)*BF - y(k).

O ganho exponencial apresenta a característica de acelerar o ajuste quando a diferença entre a referência e a saída da planta for grande.

Em [5] é apresentado um método de treinamento baseado em uma perturbação aplicada simultaneamente a todos os pesos da rede, procedimento similar aplicado em implementação de hardware e fabricantes de circuitos analógicos VLSI para redes neurais. O método consiste em adicionar de forma randômica um valor entre -0,01 a -0,001 e 0,001 a 0,01, de acordo com o exemplo citado, além de um fator multiplicador. O erro analisado é a saída da planta e não do controlador, ou seja, saída atual da planta menos a saída desejada da planta (referência). A grande dificuldade está em se estabelecer qual é o valor adequado do multiplicador e se estabelecer as faixas acima citadas para cada aplicação. Por ser um método de gradiente estocástico, pode haver dificuldade de convergência para valores pequenos de erro. Deve-se considerar ainda a preocupação com a amplitude da perturbação que pode facilmente levar o treinamento a uma direção errada quando da atualização dos pesos, visto que todos os pesos são "perturbados" durante a iteração.

Consideramos que o método proposto neste trabalho para a realização de controladores neurais diretos com o desconhecimento do modelo do processo é mais aplicável de uma forma genérica que o mencionado em [5].

6. Testes do Ambiente no Controle de uma Planta Piloto

6.1 Descrição da Planta Piloto

Os testes para verificação do comportamento dos modelos de controle foram feitos utilizando-se um gerador de 127 VCA de 1800 VA impulsionado por um motor monofásico.

O campo do gerador foi controlado através de uma ponte hexafásica constituída de SCRs. É importante salientar que além do comportamento tipicamente não-linear do processo controlado, a ponte hexafásica também possui um comportamento não-linear respondendo ao seno do sinal aplicado.

Um varivolt ou autotransformador foi utilizado para provocar perturbações ao controlador, bem como, para permitir o condicionamento do nível de tensão adequado a ser aplicado a bobina de campo do gerador através da ponte de tiristores, conforme ilustrado na figura 20.

Durante o estudo de caso foi utilizada a planta da figura 20. Para tanto, não tem-se disponível o modelo matemático ideal da planta. A figura 21 ilustra a relação saída/entrada da planta com o intuito de mostrar o aspecto não-linear ao longo da faixa de atuação para que foi projetado o controlador neural.

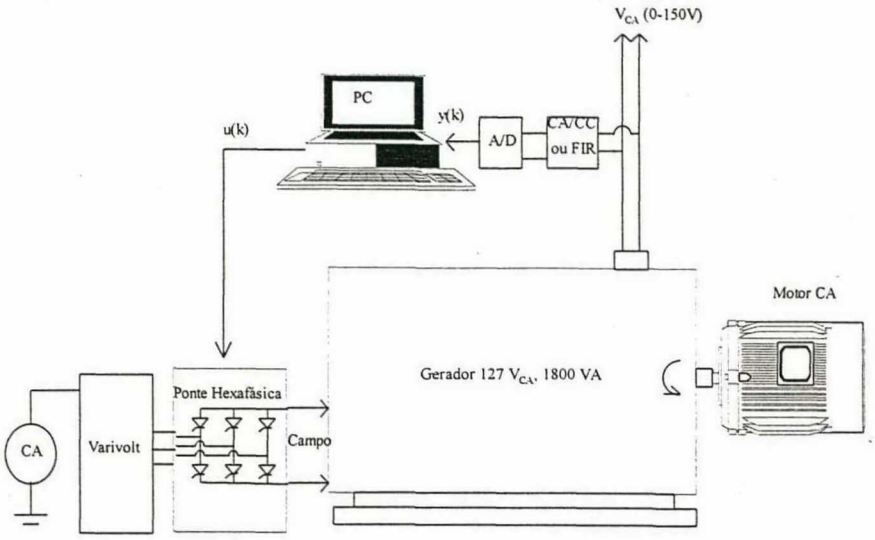


Figura 20 - Planta piloto: gerador

A determinação do modelo matemático aproximado da planta analisada foi feita aplicando-se um sinal degrau considerando-a como um processo de 1ª ordem.

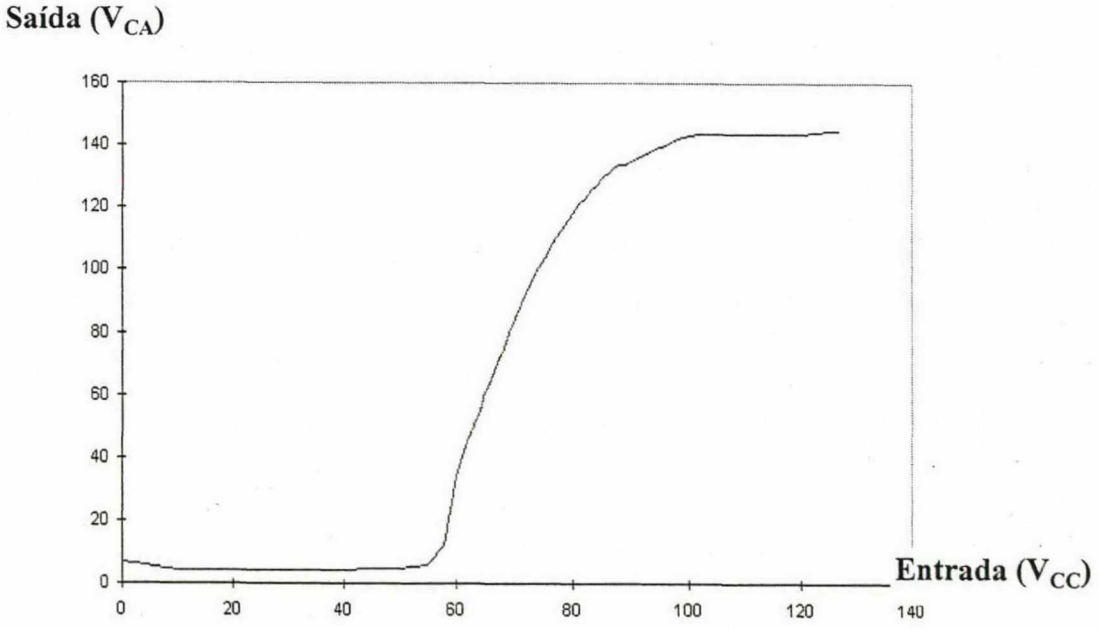


Figura 21 - Relação saída/entrada da planta

Com relação ao gerador, o modelo determinado em 115 V_{CA} que corresponde o valor da tensão nominal do gerador ficou como:

$$f_{gerador}(s) = \frac{1,5}{1 + s \times 0,35} \tag{46}$$

Para o transdutor utilizado, o modelo equivalente foi o seguinte:

$$f_{transdutor}(s) = \frac{30,0}{1 + s \times 0,55} \tag{47}$$

6.2 Testes Realizados e Resultados Obtidos

Os testes foram feitos comparando-se o desempenho do controlador neural em relação a um PID discreto clássico previamente dimensionado para este gerador/motor.

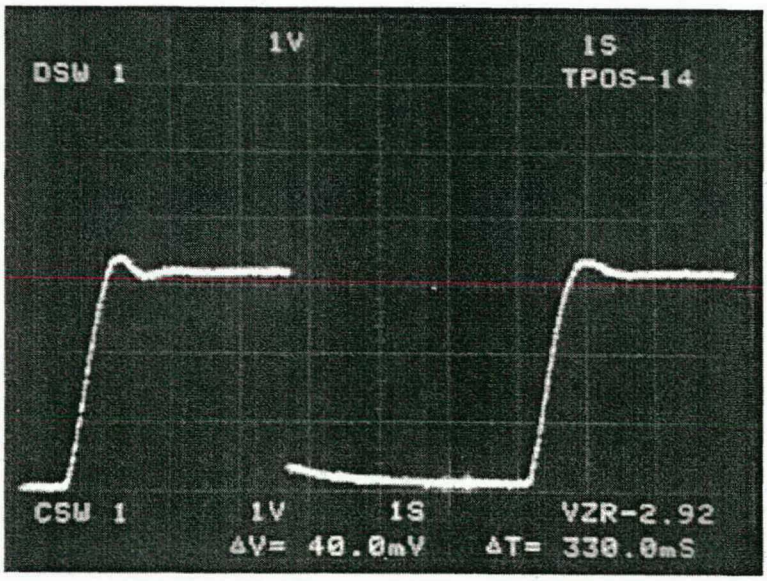


Figura 22 - PID & Controlador Neural

Na figura 22 são mostradas a forma de onda obtida diretamente da tela do osciloscópio onde são comparados o desempenho a degrau de um controlador PID discreto (forma de onda da esquerda do observador) e o desempenho do controlador neural (forma de onda da direita). Este sinal foi obtido somente utilizando os "true vectors", mediante a variação da velocidade. No caso, foi usada velocidade igual a 20. Velocidades inferiores a 20 aparentemente apresentam uma resposta melhor ao ensaio ao degrau, somente que o

desempenho passou a ser inferior ao do PID para perturbações provocadas diretamente sobre o processo, perturbações que foram aplicadas no varivolt (autotransformador). Neste ensaio, os resultados são válidos tanto para o controle direto como indireto, visto o treinamento ter sido feito somente utilizando os "true vectors". A rede controladora utilizada foi de duas camadas intermediárias, com respectivamente 5 e 3 neurônios por camada.

No ensaio seguinte (figura 23), foi variada a velocidade ("true vectors") para 30, 70 e 300 respectivamente. Neste teste foram utilizadas duas redes em paralelo de mesma dimensão, com 2 camadas intermediárias com 3 neurônios em cada uma. A faixa de atuação da primeira rede foi definida entre 0 e 55 V. Já a segunda, entre 45 e 150 V.

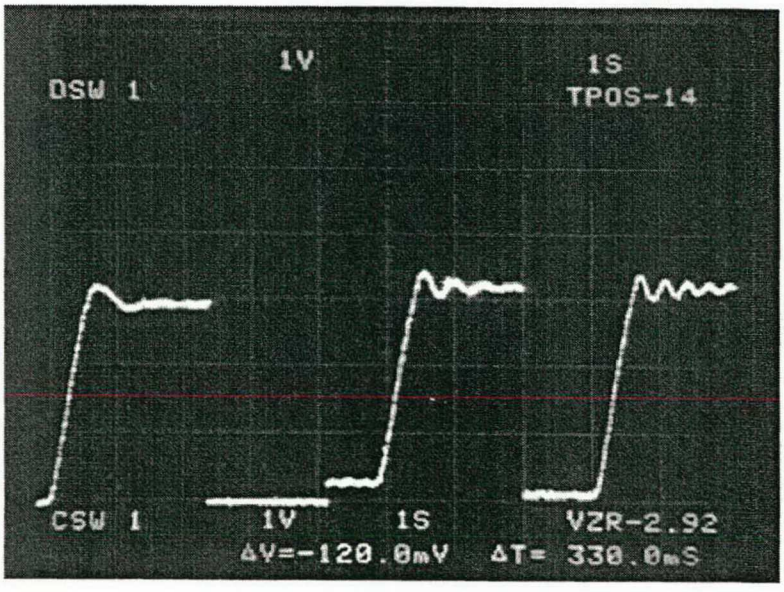


Figura 23 - Efeito da variação da velocidade

Observar que na forma de onda central aparece o efeito resultante do mau dimensionamento do conjunto de treinamento, afetando o treinamento da rede da primeira faixa. Este efeito foi conseguido retirando-se um dos vetores de treinamento exclusivamente para o ensaio com velocidade 70.

No ensaio seguinte é apresentado o efeito da captura dinâmica, variando-se o "delay accelerator". Na forma de onda da esquerda é mostrado o resultado do treinamento somente com os "true vectors". Neste ensaio, não foi observada variação significativa no comportamento para o controle direto e indireto. A diferença reside normalmente no tempo de treinamento.

Esta opção é bastante útil para se verificar em tempo real se a rede capturou corretamente o comportamento dinâmico do processo. O resultado deste ensaio é mostrado na figura 24.

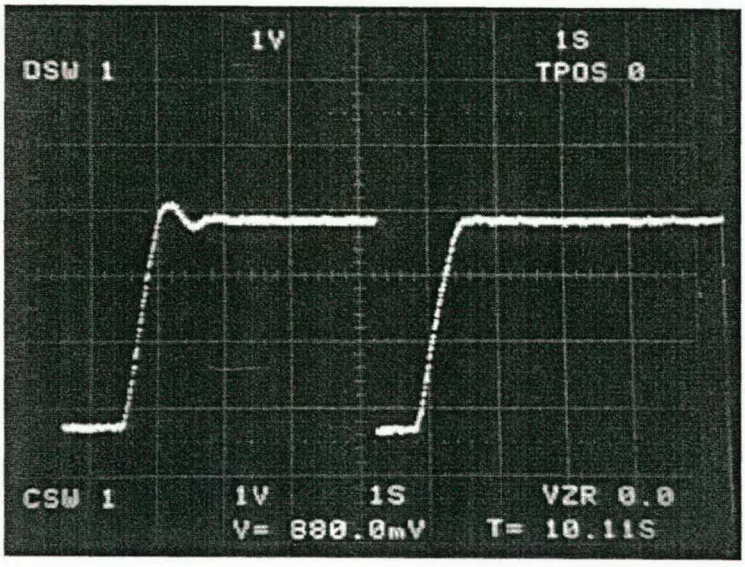


Figura 24 - Efeito "delay accelerator"

No ensaio da figura 25 tem-se o efeito da variação de velocidade aplicado sobre a rede após esta já ter sido treinada. Pode-se obter comportamento tanto subamortecido como sobreamortecido.

O modo de controle indireto necessita durante o treinamento dinâmico, da rede neural identificadora do processo, como já mencionado nos capítulos anteriores. Na figura 26 tem-se o resultado do ensaio comparativo entre o comportamento do processo observado em tempo real e a rede identificadora de uma camada intermediária com três neurônios (a) e com duas camadas intermediárias também com três neurônios em cada uma destas camadas (b).

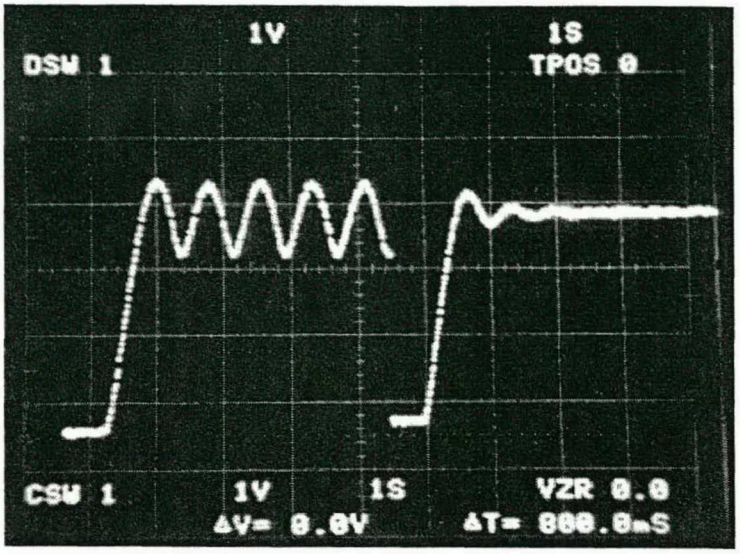
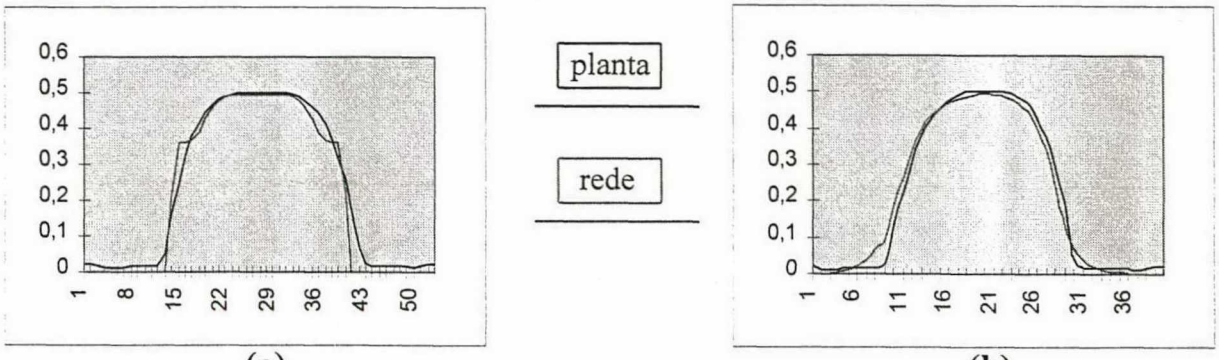


Figura 25 - Variação da velocidade pós-treinamento



(a) **(b)**

Figura 26 - Desempenho da rede identificadora em relação a planta

6.3 Conclusões dos Ensaios

Observou-se que após um tempo inicial de treinamento consegue-se estabelecer um controle mais contínuo nos extremos da faixa controlada utilizando o controlador neural para o caso do processo descrito no capítulo 6. Naturalmente isto é devido a grande não-linearidade deste processo principalmente para valores baixos de tensão.

Outra observação é que, durante a fase de treinamento, o modelo indireto converge mais rapidamente que o modelo de controle direto. O que ocorre em

contrapartida, o modelo direto necessita de menor esforço computacional por iteração, bem como menor área de memória.

O sucesso ou fracasso da realização de controladores neurais está intimamente ligado a qualidade dos vetores de treinamento utilizados e ao teste que se submete a(s) rede(s) em questão. No treinamento preliminar conseguiu-se um resultado satisfatório após 20.000 iterações somente usando o treinamento de propagação reversa e este número reduziu-se sensivelmente quando se utilizou o treinamento genético combinado.

Na plataforma utilizada, um micro computador IBM PC 386 DX 40 MHz, para uma rede com duas camadas internas com até 5 neurônios cada, resultou num tempo de treinamento inferior a 15 minutos. Nos treinamentos subsequentes, visto a rede já estar pré-treinada, este tempo se reduz a alguns segundos, dependendo do número de vetores utilizados durante o treinamento.

A cada instante, pode-se ter uma rede controladora neural em tempo real controlando a planta e uma outra independente desta, sendo treinada. Encerrado o treinamento e sendo este com sucesso, os pesos da rede controladora em tempo real são rapidamente atualizados sem interferir no processo controlado.

Após a atualização dos pesos pode-se encerrar o treinamento ou reiniciar outro de uma forma automática seletiva ou não.

O Ambiente oferece condições para a operação adaptativa por retreinamento e por acompanhamento em tempo real do erro em regime. Isto amplia a utilização do Ambiente além do ensaiado no estudo de caso. O critério de adaptação por retreinamento é feito provocando-se o disparo automático de uma captura de novos vetores de treinamento toda vez que se encerra um treinamento, sendo que os "true vectors" são sempre preservados. O procedimento para avaliação do erro em regime é detalhado no anexo B item 7.

7. Conclusão

Neste trabalho são descritos os aspectos teóricos que embasaram a formalização do problema central aqui apresentado, que visa o desenvolvimento e implementação de um ambiente para realização de controladores neurais adaptativos. São abordados os detalhes necessários para o perfeito treinamento e a correta captura dos vetores que são utilizados para o treinamento propriamente dito. Foram feitas considerações sobre as principais preocupações que deve-se ter ao longo de todo o projeto do controlador neural. São também apresentados os resultados práticos onde pode-se concluir a real aplicabilidade da solução proposta na maioria dos problemas encontrados em controle de processos. O trabalho não se resume a descrever métodos de elaboração de controladores neurais, pois culmina no completo desenvolvimento do hardware e software necessários para a implementação prática dos controladores neurais. A aplicação do Ambiente não se limita somente ao caso da planta piloto utilizada durante os testes, visto o Ambiente ser configurável por software de acordo com a topologia necessária para cada caso.

Sem dúvida, dentro da área de controle, a utilização de controladores neurais se estende além dos controladores para reguladores de tensão e velocidade de geradores. É importante salientar que para cada aplicação haverá uma estrutura mais adequada, e a mesma só será identificada após a realização de testes. O que se sugere é começar utilizando uma rede bastante simples com uma camada interna e não menos de três neurônios nesta camada. A partir dos resultados observados pode-se ir aumentando a estrutura interna da rede. Observe que, se fizermos uma partição adequada do problema, ou seja, especializando uma rede para cada ponto crítico de controle, cada rede terá estrutura mais simplificada, facilitando desde o treinamento quanto a resposta efetiva do controlador.

A grande motivação para se utilizar controladores neurais reside no fato de se poder realizar controladores até mesmo complexos, utilizando-se uma abordagem menos formal que as usuais para estes tipos de problema e ao mesmo tempo mais intuitiva.

As redes neurais são antes de mais nada aproximadores universais. Deve-se levar em conta que quanto se implementa um controlador clássico de maneira prática, também se faz num ambiente onde as informações recebidas (dos transdutores) já apresentam uma certa quantidade de erro embutida. No caso dos controladores neurais, como as informações são obtidas diretamente do processo, os erros são automaticamente considerados, inclusive aqueles implícitos do processo que não se tem pleno conhecimento ou não foram considerados.

Alguns detalhes que fazem parte deste trabalho são contribuições inéditas do ponto de vista científico: 1) Queremos destacar o método de controle direto apresentado neste trabalho; 2) o ajuste automático da taxa de aprendizado do algoritmo "backpropagation"; 3) o critério de elitização empregado no algoritmo genético; 4) o mecanismo de contorno para pequenos valores de derivada do algoritmo "backpropagation"; 5) os conceitos paralelos controlador PID/neural, tais como, ganho/velocidade e derivador/"delay accelerator"; 6) o método variante do algoritmo de "simulated annealing", entre outros, que na grande maioria, são mecanismos de concepção não complexos e no entando apresentam resultados satisfatórios.

Podemos afirmar em face ao exposto até aqui, que controladores neurais se apresentam realmente como uma opção para a grande maioria dos problemas de controle, sendo importante que em cada situação particular se explore as vantagens e desvantagens de se utilizar ou não esta técnica. A maior dificuldade normalmente reside no fato de que, durante a fase de treinamento, o usuário deve dispor de uma metodologia apropriada para obtenção dos vetores para o treinamento. É justamente neste item que o Ambiente proposto visa reduzir o esforço do usuário, fornecendo o suporte para obtenção dos dados para posteriormente serem utilizados no treinamento propriamente dito.

Sugestões para Continuidade do Trabalho

No Ambiente foram implementadas várias opções que tem como objetivo facilitar o aprendizado a partir de captura on-line e automática. Neste ínterim, houve uma relativa dificuldade em simular um processo variante no tempo, visto o estudo de caso ter sido feito sobre um processo real, e os resultados

foram obtidos de modo direto da planta sem ser por simulação. Os ensaios foram feitos exclusivamente por alteração no comportamento da ponte de tiristores, ora por inversão de seqüência de disparo, ora por remoção de um dos tiristores. Tipicamente o que se espera de um controlador adaptativo é que ele atue dentro de uma faixa, faixa esta que não cubra nenhum problema catastrófico como os descritos. Porisso, sugerimos que seja indicado algum processo mais flexível para que seja ensaiada a capacidade de adaptação do Ambiente, processo este que seja variante no tempo, com características bem conhecidas.

Num outro enfoque, acha que soluções baseadas em redes plásticas devam ser mais exploradas, quem sabe em continuidade à proposta de [45] (rede funcional + polinomial). Considera-se que, em trabalhos que envolvem controle de processo, não se pode deixar de inferir condições que permitam avaliar o efeito de variação da velocidade de resposta, tal como ocorre nos PIDs tradicionais. Em processos práticos, na quase sua totalidade, não se tem um modelo matemático perfeitamente ajustado, o que dificulta até mesmo a construção de controladores discretos. Criar um método de variação do comportamento dinâmico após captura dos dados de treinamento para uma rede do tipo funcional [2,3], considera-se que seja um desafio cujos resultados sem dúvida são de grande interesse para os pesquisadores da área.

Controle neural adaptativo pode se tornar uma panacéia senão tiver estabelecido mecanismo adequado de captura de dados e de teste. Isto porque, como é conhecido, o comportamento de uma rede neural só pode ser totalmente avaliado após testes. Se permite que uma rede capture dados por seu livre arbítrio, se auto treine e conecte-se diretamente ao processo a controlar, é muito difícil precisar se o treinamento ocorreu a contento, ou seja, se o desempenho do sistema realmente melhorou após o retreinamento. Isto ocorre para controladores inteiramente neurais como híbridos. No caso de parcialmente neural, ou seja, somente o elemento que determina a alteração de parâmetros de um controlador principal for neural, pode-se extrapolar as mesmas regras para o controlador totalmente neural, pois neste último, não é necessário que todo o controlador (neural) tenha que ser retreinado durante a fase adaptativa. Foram desenvolvidos no Ambiente duas formas de buscar garantir a memória da rede, que são, o controle de variação dos pesos e os "true vectors". No caso do controle de variação dos pesos, sugere-se desenvolver um critério mais

inteligente em vez simplesmente de limitar o máximo valor de cada peso por um critério de percentualidade.

Outro ponto significativo é a constante busca de algoritmos de treinamento mais eficientes e rápidos, mais próprios para controle em tempo real, bem como, métodos de avaliação de desempenho e robustez do controlador e também métodos de análise de estabilidade .

8. BIBLIOGRAFIA

[1]
SILVA, L. E. Borges da; TORRES G. Lambert; SATURNO, E. C. ; SILVA, A. P. Alves da; OLIVER G. - "Neural Net Adaptive Schemes for DC Motor Drives", IEEE Industry Applications Society Conference, Toronto, October 1993.

[2]
PAO, Yoh-Han. "Adaptive pattern recognition and neural networks"., Addison-Wesley Publishing Company, United States of America, 1989.

[3]
MASTERS, Timothy. "Practical neural networks recipes in C++". Academic Press, INC, San Diego CA, 1993.

[4]
McCULLOCH, W. S.; PITTS, W. H. - "A logical calculus of ideas immanent in nervous activity". Bull Math Biophys, 5:115-133. 1943. Formal Neuron.

[5]
KANATA, Yakichi; MAEDA, Yutaka. "Learning rule of neural networks for control ", SICE, 777 - 789. 1994.

[6]
BOSE, Bimal K. "Expert system, fuzzy logic, and neural network applications in power electronics and motion control". Proceedings of IEEE, vol. 82, no. 8, 1303 - 1323. 1994.

[7]
TAKAHASHI, Hiroki; AGUI, Takeschi; NAGAHASHI, Hiroshi. "Designing adaptive neural networks architectures and their learning". Science of Artificial Neural Networks II, SPIE vol. 1966. 208 - 215. 1993.

[8]
SHEBLÉ, Gerald B.; MAIFELD, Timothy T. "Unit commitment by genetic algorithm and expert system". Eletric Power Systems Research 30. 115 - 121. 1994.

[9]
WU, Q. H.; HOGG, B. W.; IRWIN, G. W. "A neural network regulator for turbogenerators". IEEE Transactions on Neural Networks. vol 3, no. 1, 95 - 100. Jan 1992.

[10]
TORRES, Germano L. "Notas do curso de introdução às redes neuronais". EFEI. 1992.

[11]
SEPEDA FILHO, Idmilson H.; STEMMER, Marcelo R. "Redes Neurais". Notas Internas LCMI/UFSC. Set/1993.

[12]
RUMELHART, David E.; LEHR, Michael A.; WIDROW, Bernard. "Neural networks: applications in industry, business and science". Communications of the ACM. Vol. 37, no. 3. 93 - 105. March 1994.

[13]
DJUKANOVIC, M.; SOBAJIC, D. J.; PAO, Y. H. "Neural net based determination of generator-shedding requirements in electric power systems". IEE proceedings-C, Vol. 139, No. 5, 427 - 436, Sep/1992.

[14]
CAMPAGNA, David P.; KRAFT, L. Gordon. "A comparison between CMAC neural network control and two traditional adaptive control systems". IEEE Control Systems Magazine. 36 - 43. April/1990.

[15]
ROY, Serge. "Near-optimal dynamic learning rate for training back-propagation neural networks". SPIE Vol. 1966 Science of Artificial Neural Networks II. 277 - 283. 1993.

[16]
JANAKIRAMAN, J.; HONAVAR V. "Adaptive learning rate for increasing learning speed in backpropagation networks". SPIE Vol. 1966 Science of Artificial Neural Networks II. 225 - 235. 1993.

[17]
CHANG, C. S.; SRINIVASAN, D.; LIEW, A. C. "A hibrid model for transient stability evaluation of interconnected longitudional power systems using neural networks/pattern recognition approach". IEEE Transactions on Power Systems. Vol. 9. No. 1, 85 - 92. Feb. 1994.

[18]
MISTRY, Sanjay I.; NAIR, Satish S. "Identification and control experiments using neural designs". IEEE Control Systems. 48 -56. June/1994.

[19]
VILLALOBOS, Leda; MERAT, Francis L. "Optimal learning capability assessment of multicategory neural nets". SPIE Vol. 1966 Science of Artificial Neural Networks II. 384 - 395. 1993.

[20]
ZHANG, Y.; CHEN, G. P.; MALIK, O. P.; HOPE, G. S. "An artificial neural network based adaptive power system stabilizer". IEEE Transactions on Energy Conversion. Vol. 8. No. 1. 71 - 77. March/1993.

[21]
WEERASOORIYA, S.; EL-SHARKAWI, M. A. "Laboratory implementation of neural network trajectory controller for a dc motor". IEEE Transactions on Energy Conversion. Vol. 8. No. 1. 107 - 113. March/1993.

[22]
DJUKANOVIC, M.; SOBAJIC, D. J.; PAO, Y. H. "Preliminary results on neural net based simulation of synchronous machine dynamic response". Electric Power Systems Research, 25. 159 - 168. 1992.

[23]
YANG, H. T.; HUANG, K. Y.; HUANG, C. L. "An artificial neural network based identification and control approach for the field-oriented induction motor". Eletric Power Systems Research, 30. 35 - 45. 1994.

[24]
OWEN, C. B.; ABUNAWASS, A. M. "Application of simulated annealing to the backpropagation model improves convergence". SPIE Vol. 1966. Science of Artificial Neural Networks II. 269-275. 1993.

[25]
HESKES, T.; WIEGERINCK, W.; KOMODA, A. "Scaling properties of on-line learning with momentum". IEEE International Conference on Neural Networks. IEEE World Congress on Computational Intelligence, Orlando, FL, USA, p.508-12. Vol 1. 27 June-2 July 1994.

[26]
TANOMARU, J.; "Motivação, fundamentos e aplicações de algoritmos genéticos". Proc. II Congresso Brasileiro de Redes Neurais. Curitiba, PR, Brasil, 29 outubro - 01 novembro 1995.

[27]
ALMEIDA, R. A.; PIAZZAROLLO, M. S. "Identificação e controle de um processo térmico utilizando redes neurais artificiais com memória". Proc. II Congresso Brasileiro de Redes Neurais. Curitiba, PR, Brasil, 29 outubro - 01 novembro 1995.

[28]
SASTRY, P. S.; SANTHARAM, G.; UNNIKRISHNAN, K. P. "Memory neuron networks for identification and control of dynamical systems". IEEE Transactions on Neural Networks. Vol. 5. N° 2, p. 306-19. March 1994.

[29]
KOIVISTO, H.; RUOPPILA, V. T.; KOIVO, H. N. "Real-time neural networks control-an IMC approach". Proceedings of the 12th Triennial World Congress o International Federation of Automation Control. Vol. 4. Sydney, NSW, Australia, 18-23 July 1993, p. 127-32.

[30]
PARK, S.; PARK, L. J.; PARK, C. H. "A neuro-genetic controller for nonminimum phase systems". IEEE Transaction of Neural Nets, vol. 6, no. 5, p. 1297-300. Sept. 1995.

[31]
ROBAINAS, R. R. D.; PANDYA, A. S. "An application of backpropagation neural architectures to the realization of control transfer functions and compensators". Proc. SPIE, vol. 2243, p. 370-9. Orlando, FL, USA, 5-8 April 1994.

[32]
KLINGUELFUS, M. C.; PAGANO, D.; STEMMER, M. R. "Modernização de reguladores de velocidade e tensão para hidrogeradores". VI Encontro Regional Latino-americano da CIGRÉ. p. 01-06. Foz do Iguaçu, PR, Brasil, 28 maio - 01 junho 1995.

[33]
OH, S. Y.; KIM, H. G. "Stable neural controller design: composite adaptive learning and hybrid control architecture". Trans. Korean Inst. Electr. Eng., vol. 44, no. 4, p. 508-17. April 1994.

[34]
DAMLE, R.; RAO, V.; KERN, F. "Multivariable neural network based controllers for smart structures". J. Intell. Mater. Syst. Struct, vol. 6, no. 4, p. 516-28. Rolla, MO, USA.

[35]
CHEN, F. C.; CHANG, C. H. "Practical stability issues in CMAC neural network control systems". Proceedings of the 1995 American Control Conference, p. 2777-81, vol. 4. Seattle, WA, USA, 21-23 June 1995.

[36]
DELGADO, A.; KAMBHAMPATI, C.; WARWICK, K. "Dynamic recurrent neural network for system identification and control". IEE Proc. Control Theory Appl., vol. 142, no. 4, p. 307-14. UK. July 1995.

[37]
MISTRY, S. I.; NAIR, S. S. "Real-time experiments in neural identification and control of disturbance". Proceedings of the 1995 American Control Conference. Seattle, WA, USA, p. 2307-11, vol. 3. 21-23 June 1995.

[38]
LIPPE, W. M.; FEURING, T.; TENHAGEN, A. "A hybrid learning rule for a feedforward network". Third Golden West International Conference. Las Vegas, NV, USA. 6-8 June 1994. p. 13-18, vol. 1.

[39]
BUCZAK, A. L.; UHRIG, R. E. "Information fusion by fuzzy set operation and genetics algorithms". Simulation Councils, Inc. Vol. 65. No. 1, p. 52-66. July 1995.

[40]
RIEDMILLER, M. "Advanced supervised learning in multi-layer perceptrons from backpropagation to adaptive learning algorithms". Special Issue on Neural Networks, Int Journalon Computer Standarts and Interfaces, 1994.

[41]
BRAUN, H.; RIEDMILLER, M. "A direct adaptive method for faster backpropagation learning: the RPROP algorithm". Proceedings of the IEEE International Conference on Neural Networks. San Francisco, March 28 - April 1, 1993.

[42]
FREEMAN, James A. "Neural networks: algorithms, applications, and programming techniques". Addison-Wesley Publishing Company, United States of America, 1991.

[43]
LAWRENCE, Jeannette. "Introduction to neural networks and expert systems". California Scientific Software. United States of America, 1992.

[44]
RUMELHART, David E.; McCLELLAND, James L. "Parallel Distributed Processing." Vol. 1: Foundations. The MIT Press, 1969. Third Printing, 1988.

[45]
NASCIMENTO, P. C.; ALVES DA SILVA, A. P.; LAMBERT TORRES, G. "Uma rede neural plástica para controle em tempo-real". II Simpósio Brasileiro de Automação Inteligente. P. 277-82. Curitiba. Brasil. 1995.

[46]
LIPPMAN, Richard P. "An introduction to computing with neural nets". IEEE ASSP Magazine. April 1987.

[47]
BARTO, A. G. "Connectionist learning for control: an overview". COINS Technical Report 89-89. University of Massachusetts, Amherst, MA, USA. 38 p., 1989.

[48]
SANNER, R. M.; SLOTINE, J. J. E. "Gaussian Networks for Direct Adaptive Control". Massachusetts Institute of Technology. Cambrige, MA, USA. 39 p., 1991.

[49]
DRAEGER, A.; ENGELL, S.; RANKE, H. "Model predictive control using neural networks". IEEE Control Systems. P. 61 -5October 1995.

[50]
STEMMER, M. R.; KLINGUELFUS, M. C. "Implementation and test of an integrated environment for adaptive control". 13 th International Conference on Applications of Intelligent Software Systems in Power Plant, Process Plant and Structural Engineering. 8 p. São Paulo. Brasil. 1995.

[51]
STEMMER, M. R.; KLINGUELFUS, M. C. "Implementação de um ambiente integrado para controle neural adaptativo". II Congresso Brasileiro de Redes Neurais. 7 p. Curitiba. Brasil. 1995.

[52]
STEMMER, M. R.; KLINGUELFUS, M. C. "Implementation and test of an integrated environment for adaptive neural control". Second Latin American Seminar on Advanced Control - LASAC'95 and Fourth Seminar on System Identification, Parameter Estimation and Adaptive Control - SISEPCA'95. 7 p. Santiago. Chile. 1995.

ANEXO A

Características gerais do Ambiente

- Não requer modelo matemático do processo tanto no controle direto como indireto.

- Treinamento OFF-LINE/ON-LINE e controle ON-LINE simultâneo.

- Aquisição ON-LINE de novos vetores de treinamento.

- Permite a definição de redes especializadas para diferentes regiões de operação do sistema.

- O comportamento adaptativo pode ser obtido nas seguintes condições:
  - Durante o próprio treinamento;
  - Através de múltiplas redes especializadas;
  - Em função do erro de regime:
    - fixação do valor de saída.
    - selecionamento de outro controlador neural.
    - retreinamento.
  - Recursos para captura/treinamento com disparo automático ou disparo externo tais como:
    - filtros de captura.
    - variação do sinal de controle automaticamente.

- Opera em microcomputadores tipo IBM PC 386 ou compatível com co-processador aritmético ou superior, ambiente DOS 5.0 ou versão mais recente, com no mínimo 4 megabytes de memória RAM.

- Algoritmo de treinamento: genético, "backpropagation", "backpropagation" + "simulated annealing", ou híbrido.

- Tempo médio de treinamento: de 10 s a 15 min. (para o estudo de caso analisado).

- Ajuste automático dos principais parâmetros de treinamento, entre eles: taxa de aprendizagem (obtenção do mínimo efetivo da função), ganho linear e exponencial.

- Controle de não convergência do algoritmo genético através de duas opções:
  - Mutação controlada;
  - Mutação elitizada.

- Resguarda da "memória":
  - "True vectors";
  - Controle de variação dos pesos.

- Compatibilidade com os conceitos de ajuste associados ao PID clássico (ganho = velocidade; derivador = "delay-accelerator"; integrador = erro final).

- Ajuste velocidade de resposta da rede em tempo real (efeito derivativo) após a rede já ter sido treinada.

- Permite a instalação de placas de aquisição de dados de fabricantes diferentes. Hardware adaptável.

- Taxa de amostragem (tempo real) variável não necessitando de relógio externo.

- Permite o treinamento e teste de redes com propósito geral.

ANEXO B

Utilização do Ambiente

O Ambiente proposto foi implementado de maneira a evitar que erros de definição gerados pelo usuário durante a utilização do Ambiente, venham comprometer o funcionamento do programa, e principalmente, procurando evitar bloqueamentos indesejáveis de software devido a inconsistências na definição dos vários parâmetros necessários para o controlador neural. Desta forma, a cada mudança de tela é feito um teste de consistência de todas as variáveis que estiverem sendo utilizadas naquele instante.

Para utilizar o Ambiente, basta digitar na linha de comando do DOS a palavra "neu_con". Isto fará com que todos os módulos do Ambiente sejam carregados e realizada a alocação de memória. É importante que todos os arquivos de dados que serão utilizados estejam no mesmo diretório que o programa principal. Caso se deseje acionar o Ambiente a partir do "aviso do MS-DOS" do Windows, deve-se tomar o cuidado de acionar o Windows do diretório que contém o arquivo neu_con.exe, bem como, todos os arquivos de dados que serão utilizados durante a execução do Ambiente. Caso o Windows seja disparado por outro diretório, durante a execução normal do Ambiente será enviado mensagens de que os arquivos de dados não podem ser abertos. Isto se deve a restrições do próprio Windows.

Na tela inicial da figura 27, aparecem 4 opções ou modos de operação que podem ser selecionados cada um escolhendo a letra de destaque associado ao respectivo modo:

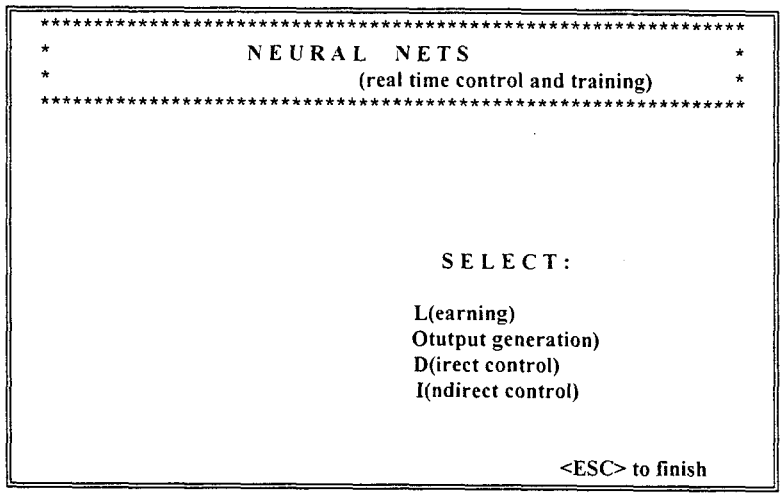


Figura 27 - Tela inicial

O modo *L(earning)* se refere ao treinamento supervisionado típico encontrado na maioria dos ambientes de desenvolvimento de redes neurais existentes, onde através de um conjunto de vetores de treinamento, pode-se construir uma determinada rede neural, independente da aplicação.
O modo *0(utput generation)* permite que seja realizado os testes sobre alguma rede já construída, como por exemplo, a resultante do modo L(earning).
O modo *D(irect control)* realiza um controlador neural usando o método de controle direto.
O modo *I(ndirect control)* realiza um controlador neural usando agora o método de controle indireto.

Em seguida destacaremos cada um destes quatro modos de operação:

B.1 MODO L(earning)

Este modo foi implementado com o intuito de tornar o Ambiente de propósito geral.

Em todas as 4 opções ("learning", "output generation", controle direto e controle indireto), temos os dados ou variáveis registrados em um arquivo de estrutura denominado de "task file" cuja terminação é "_v.dat".

Na tela da figura 28, em primeiro lugar devemos definir um nome de um arquivo de estrutura (*"task name"*). Se já existe algum arquivo de estrutura anterior, basta digitar o nome deste arquivo. Observar que a terminação "_v.dat" é inserida automaticamente não devendo ser digitada. Caso seja simplesmente digitado a tecla "ENTER", será carregado o último arquivo de estrutura utilizado. Se ainda for digitado "?", será apresentado na tela todos os arquivos do diretório corrente com terminação ".dat". O caracter "?" pode ser usado para os arquivos de dados de pesos ("weight data file") e de vetores de treinamento ("input data file") tanto neste como nos outros modos.

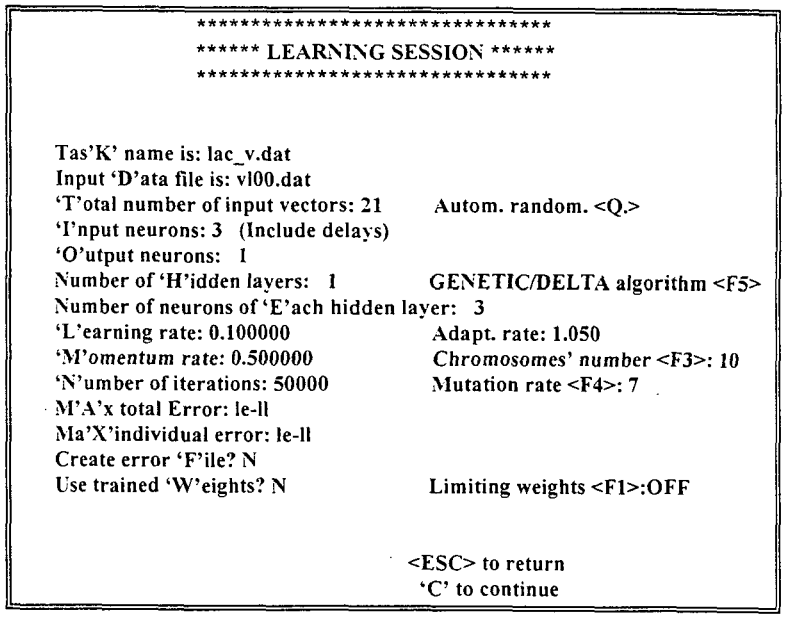


Figura 28 - Treinamento off-line

Na linha seguinte *"Input Data file"* deve ser digitado somente o nome de um arquivo de dados, sem extensão, pois será inserida automaticamente. Este arquivo deve conter todos os vetores de treinamento. Cada vetor deve ocupar uma linha, sendo que as primeiras colunas deverão corresponder as entradas e por último, as saídas. O arquivo de dados poderá conter tanto dados normalizados como não. Para o caso em que os dados forem não normalizados, devemos ter nas duas primeiras linhas respectivamente o valor limite inferior e

o valor limite superior referente a cada coluna. A partir deste ponto a seleção será feita de maneira automática.

Em "*Total number of input vectors*" devemos indicar quantos vetores de treinamento queremos que sejam lidos do arquivo correspondente. Observar que para o caso dos vetores não normalizados, não devemos considerar as duas primeiras linhas que correspondem aos valores limites.

"*Autom. random.*" é uma abreviação de randomização automática aqui também denominada de randomização controlada. Este modo está associado com o algoritmo de treinamento baseado no método "backpropagation + simulated annealing". No Ambiente proposto a definição das temperaturas é feita de maneira automática. É necessário somente que sejam definidos o número máximo de randomizações. Em seguida deve-se definir qual deverá ser o valor mínimo da taxa de aprendizagem ("learning rate") a partir do que será disparada uma nova randomização.

As linhas seguintes correspondente ao número de neurônios de entrada e saída, número de camadas intermediárias ou "hidden" e o número de neurônios de cada camada intermediária. Estas linhas devem ser preenchidas de acordo com o conhecimento que o especialista tem do problema que esta sendo analisado.

A opção "*GENETIC/DELTA*" permite que seja escolhido qual dos algoritmos de treinamento será utilizado. A opção DELTA ou "backpropagation" inclui o "simulated annealing". Caso seja escolhido um dos dois algoritmos em vez dos dois, serão mantidos na tela somente os parâmetros relativos ao algoritmo selecionado.

A taxa de aprendizagem ("*learning rate*") deve ser preenchida com o valor que será utilizado no início do treinamento "Delta Rule". Este valor será divido pelo valor denominado como "*adapt. rate*", abreviação de "adaptive rate", toda vez que houver uma não convergência do valor do erro durante o treinamento. Deste modo, procura-se obter o valor mínimo da função, visto que o valor da taxa de aprendizagem poderá atingir um valor tão pequeno quanto se deseje.

O valor da taxa de momentum ou "*momentum rate*" não sofre nenhuma alteração de maneira automática, devendo ser definida como um valor menor que 1 (um).

Em relação ao treinamento genético, devemos definir o número de cromossomos ("*chromosomes number*") e a taxa de mutação ("*mutation rate*").

O número de cromossomos determina o tamanho da população inicial, ou seja, quantos conjuntos de vetores de pesos serão tratados a cada ciclo do algoritmo, e a taxa de mutação determina quantas mutações na população ocorrerão a cada ciclo do algoritmo genético.

No item número de iterações ("*number of iterations*") são definidos tanto o número máximo de iterações no algoritmo genético como no "backpropagation" de forma independente.

"*Max. total error*" se refere ao máximo erro aceitável para todo o conjunto de vetores de treinamento e "*max. individual error*" se refere, por conseqüência, ao erro máximo aceitável para cada um dos vetores de treinamento. O treinamento será encerrado com sucesso quando um dos dois limites for atingido.

A opção "*create error file*" permite que seja salvo em um arquivo de dados cujo nome é "criter.dat" o valor do erro total de cada iteração.

"*Use trained weights*" é usado quando se deseja utilizar algum conjunto de pesos já existentes. Para isto basta indicar somente o nome deste arquivo. A terminação "_w.dat" será incluida automaticamente.

Já a opção "*limiting weights*" permite que seja definido uma variação máxima para os valores dos pesos em relação ao seu valor inicial. A cada iteração e a cada alteração no valor dos pesos é comparado o novo valor com o valor utilizado na primeira iteração do ciclo de treinamento. Caso este valor ultrapasse o limite preestabelecido, o valor do peso correspondente será mantido igual ao limite referente à aquele peso.

Por fim, digitando-se a tecla <ESC> retornamos a tela inicial da figura 27 e digitando a tecla <C> passamos a próxima fase.

Na fase seguinte é perguntado se deseja ou não que sejam apresentados na tela os valores que serão lidos do arquivo que contém os vetores de treinamento. Basta responder com sim (Y) ou não (N). A partir deste ponto é iniciado o treinamento propriamente dito. A parte relativa ao treinamento será apresentado no item B.8.

B.2 MODO O(utput generation)

Este modo foi implementado com o intuito de facilitar os testes da rede treinada no modo "learning" (B.1). Inicialmente é necessário que seja criado um arquivo de dados com os vetores de teste devidamente normalizados em correspondência com a rede definida na tarefa ("task") especificada. Deve-se respeitar o número de entradas e saídas, ou seja, de colunas, de maneira similar as utilizadas durante o treinamento.

A primeira questão é se o teste será realizado utilizando-se uma rede com a mesma topologia que a atualmente definida, ou seja, se será mantida a mesma tarefa ("task") ou não. Para observar detalhes sobre uma determinada tarefa, deve-se pressionar a tecla <ESC> e retornar ao modo B.1.

Continuando dentro do modo B.2, é perguntado qual o nome do arquivo de teste a ser utilizado. Definido o nome, deve-se informar quantos vetores de teste deseja-se que sejam ensaiados. Após, é apresentado na tela o resultado do teste realizado.

B.3 MODO D(irect control)

Neste modo foram implementados todos os recursos necessários para realização do controlador neural, indo deste a parte referente a captura dos vetores de treinamento obtidos diretamente do processo, bem como, o treinamento propriamente dito e a realização do controlador em tempo real.

A captura dos vetores de treinamento deve ser realiza em duas fases: Na primeira delas deve-se fazer a captura dos "true vectors" (item B.9), ou então, caso não seja possível, deve-se gerá-los estaticamente por algum outro método alternativo, utilizando principalmente o conhecimento que o especialista tem do processo. Na fase seguinte, devemos fazer a captura dinâmica de novos vetores de treinamento (item B.10). Deve-se no entanto observar, que para que seja realizado a captura dinâmica, é necessário que se realize primeiramente o treinamento da rede utilizando os "true vectors".

Serão mencionadas aqui somente as opções não tratadas no modo B.1 em correspondência com a figura 29.

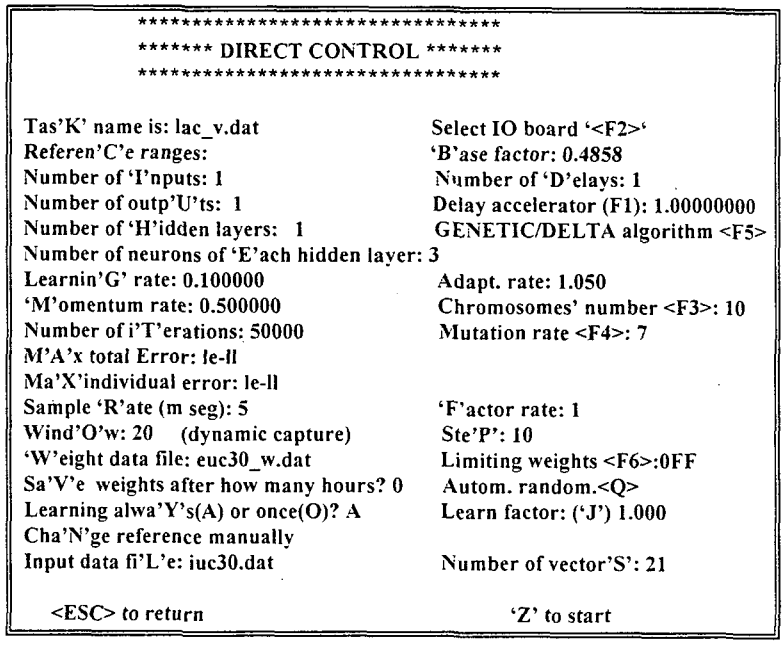


Figura 29 - Controle direto

"*Select io board*" permite que sejam utilizadas interfaces A/D (analógico/digital) de vários fabricantes, e até mesmo, uma interface fantasma ou "dummy". A interface "dummy" permite que seja ensaiado o Ambiente em plataformas que não tenham nenhuma interface A/D instalada. Dentro ainda da

opção "dummy", são provocadas variações controladas nos valores lidos, buscando simular em parte um efeito mais real de captura. A introdução de interfaces A/D não definidas no Ambiente, deve ser feita diretamente no módulo (programa fonte) "io_board.c", nos espaços apropriados para tal. Deve ser introduzida uma rotina de inicialização da nova interface, uma de leitura e/ou escrita, e por fim a opção de denominada na tela do micro. As instruções estão no próprio módulo fonte. No item B.11 são discutidos os aspectos gerais para a realização da compilação dos vários módulos fonte.

"*Reference range*" diz respeito a possibilidade de se utilizar mais de uma rede neural em paralelo num mesmo problema. Para isto, deve-se definir unicamente qual é a faixa correspondente de valores da referência do controlador que serão utilizados em cada uma das rede que atuarão em paralelo. Por exemplo, caso tenhamos uma faixa total de variação dos valores da referência entre 0 e 100, podemos definir uma rede que irá trabalhar entre 0 e 55 e outra que irá trabalhar entre 45 e 100. Desta forma, para qualquer valor instantâneo da referência, teremos ao menos uma rede neural controlando o processo. Observe, que no exemplo acima, haverá uma sobreposição entre as redes na faixa que vai de 45 a 55. Neste caso, é feito uma interpolação linear dos valores correspondentes da saída de cada uma das redes de maneira que, quando estivermos no valor 45, somente a primeira rede terá alguma contribuição e em 55 somente a segunda rede estará contribuindo com o sinal de controle. Já, exatamente em 50, teremos o sinal de controle como sendo a média aritmética entre as saídas das duas redes. Não existe a princípio um número limite de redes para um mesmo problema. Durante a elaboração desta opção foi analizada a possibilidade de se fazer com que o critério de sobreposição fosse difuso ou "fuzzy". Esta opção foi suprimida devido que, fere uma das premissas iniciais que culminaram na realização deste Ambiente, que é a de diminuir ao máximo o número de itens a serem definidos pelo usuário. A necessidade de se estabelecer variáveis lingüísticas pode em parte ser substituída pela introdução de redes especializadas com sobreposição de atuação, o que facilita o processo de fusificação e defusificação, permitindo que o usuário tenha uma perfeita idéia de como está sendo feito a ação de controle, não tendo que memorizar os limites associados as variáveis lingüísticas.

"*Base factor*" é uma constante ou uma expressão que correlaciona o valor da referência com o valor proveniente do sinal realimentado de controle, lido pela

interface A/D diretamente na saída do processo. Durante uma implementação prática de um controlador é necessário estabelecer uma relação entre o valor do sinal realimentado que é aplicado à rede neural e o sinal de referência aplicado a esta mesma rede como mostra a figura 30. Dentro da faixa de variação da referência do controlador, devemos indicar através do "base factor" qual deverá ser o valor que multiplicado pelo valor atual da referência, deve indicar a condição de erro de controle igual a zero. Para isto devemos considerar o sinal já normalizado aplicado a rede, e a relação numérica produzida pela interface A/D, bem como, as relações entrada/saída dos transdutores utilizados. Exemplificando, caso tenhamos uma variação possível da referência entre 0 e 100, e uma interface A/D que, para uma variação de 0 a 10 V, produza como saída um valor numérico entre 0 e 2047. Se a grandeza de saída do processo tiver uma variação de 0 a 200 e o transdutor acoplado entre a saída do processo e a interface A/D tiver um ganho de 0,025, teremos que quando a saída do processo estiver em 100, o valor aplicado a entrada da interface A/D será de 2,5 V, e conseqüentemente aproximadamente 511 o valor numérico na saída da interface. No caso da referência ser 50, com os limites de normalização sendo 0 e 100 para a referência, e 0 e 2047 para o sinal realimentado, o sinal aplicado a rede será de 0,5 na entrada correspondente a referência e 0,25 na entrada correspondente a entrada do sinal realimentado. Logo, se desejamos indicar esta situação como uma condição de erro nulo, devemos multiplicar o valor da referência por 0,5. Este valor 0,5 corresponde ao valor do "base factor".

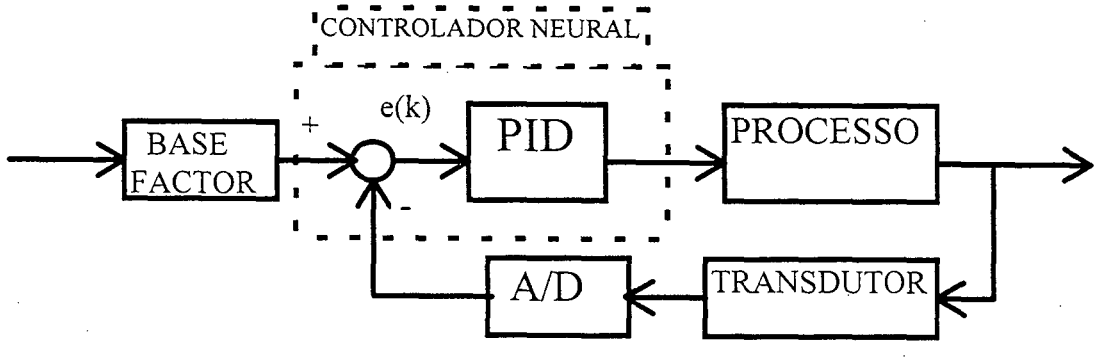


Figura 30 - Atuação do "base factor"

Uma consideração importante a ser feita diz respeito ao número de entradas, ou melhor, o que é considerado efetivamente entrada nos modos "learning" e "output generation" e o que é considerado entrada nos modos de controle direto e indireto. Nos dois primeiros modos, entrada diz respeito a todas as entradas propriamente dita. Já nos modos referentes ao controle direto e indireto, como

entrada é considerado somente as entradas de hardware existentes, em outras palavras, as entradas associadas a interface A/D. As outras entradas que são a referência e os atrasos ("*delays*") passam a ter cada uma a sua própria conotação. Isto se deve a fato da necessidade de se manter uma perfeita coerência entre os aspectos gerais do Ambiente como o linguajar típico dos profissionais da área de controle.

Tipicamente "*number of delays*" está associada a ordem do processo a ser analisado. Para um processo de primeira ordem é normalmente utilizado um único "delay", para um processo de segunda ordem é utilizado dois "delays" e assim sucessivamente. Isto de forma alguma é uma regra, mas auxilia na fase preliminar, ou seja, no anteprojeto do controlador neural. A avaliação final só poderá ser feita durante os testes finais.

"*Delay accelerator* " corresponde a uma forma indireta de se reproduzir um comportamento similar ao de um derivador quando se trabalha com controladores PIDs. Este fator é usado para multiplicar os valores do instante (k-1), ou seja, do instante de amostragem anterior ao atual. Este efeito aparecerá a segunda potência para o instante (k-2), terceira potência para o instante (k-3) e assim sucessivamente. Observe que o "*delay accelerator*" só opera sobre os atrasos ("delays").

"*Sample rate*" é taxa de amostragem aplicada ao controlador neural em tempo real, ou seja, à aquela rede que efetivamente estará controlando o processo. Esta amostragem é feita utilizando-se diretamente o relógio interno do microcomputador. Como restrição, o maior valor que se pode obter diretamente é um valor inferior a 55 ms (mili segundos). Para contornar este problema, para o caso de processos que exijam uma taxa de amostragem superior a esta, foi introduzido o "*factor rate*" que multiplica o efeito da taxa de amostragem. Por exemplo, se desejarmos uma taxa de amostragem final de 100 ms, podemos utilizar a "*sample rate*" igual a 50 e o "*factor rate*" igual a 2. O mesmo efeito será obtido fazendo o "*sample rate*" igual a 20 e o "*factor rate*" igual a 5, ou outra combinação que o produto gere o número 100. Devemos observar que a alteração do "*sample rate*" provoca alteração no relógio/calendário utilizado para amostrar a hora cronológica. Internamente ao Ambiente foi implementado uma rotina que diminui bastante este efeito. No entanto, o erro gerado será tanto menor quanto menor for o "*sample rate*" utilizado. O limite inferior do "*sample*

rate" depende do desempenho da própria plataforma de hardware utilizada. A maioria dos testes foram feitos utilizando um PC AT 386 - DX 40 MHz. Nesta plataforma foram feitos vários testes com taxa de amostragem próxima aos 500 μs utilizando duas redes em paralelo cada uma com 2 duas camadas intermediárias de 5 e 3 neurônios respectivamente por camada. O próprio Ambiente faz o controle de reentrância da rotina que é responsável pela rede em tempo real, evitando que haja bloqueio total do microcomputador, visto o DOS não permitir redenominadas de uma mesma rotina.

"*Window*" é a forma utilizada para designar janela de captura dinâmica. Ver item B.10. Quando se está realizando o treinamento a partir dos "true vectors", o valor de "*window*" deve ser mantido em zero. Caso desejemos fazer uma captura dinâmica, "*window*" deverá corresponder ao total de vetores de treinamento a serem capturados, podendo ser subdivida em blocos (item B.6).

"*Step*" também tem relação com a captura dinâmica. Ver item B.10. Se o "*step*" for igual a zero, será feito uma captura de um novo vetor de treinamento a cada "sample rate". Se o "*step*" for igual a um, será capturado um vetor de treinamento durante uma amostragem e não será feito a captura na amostragem seguinte. Se o "*step*" tiver um valor igual a dois, será capturado um vetor de treinamento e passado duas amostragens sem captura. Este ciclo se repete até que o bloco ou a janela correspondente tenha sido preenchida.

"*Save weights after how many hours*?" faz menção a um aspecto de segurança do controlador quando este estiver operando no modo adaptativo, ou seja, quando o treinamento da rede é disparado de maneira automática provocando alterações nos pesos da rede controladora que não totalmente acompanhadas pelo especialista. Esta opção permite que em intervalos regulares de tempo sejam salvados os valores dos pesos da rede controladora em um arquivo de dados correspondente. Desta maneira, mesmo após uma falha catastrófica no controlador, poderemos reinicializar o trabalho com os pesos anteriormente salvos. Deixando esta opção com o valor em zero, não ocorrerá o auto salvamento.

"*Learning always or once?*" é uma opção que está diretamente ligada ao treinamento adaptativo. Ao selecionar "*always*", o ciclo de treinamento se repetirá indefinidamente, podendo ser o disparo de maneira automática ou por

um disparo externo (item B.6). Já selecionando-se "*once*", será disparado um único ciclo de treinamento, sendo que o redisparo deverá ser manual.

A opção "*change reference*" define como ocorrerá a variação da referência do controlador, se esta será manual, automática ou baseada num vetor de valores. Maiores detalhes constam no item B.6.

"*Learn factor*" é o fator que permite que haja um contínuo aprimoramento no modo de treinamento adaptativo, quando é mantido o treinamento com disparo automático. A cada ciclo completo de treinamento com sucesso, o valor do "*learn factor*" é utilizado para reduzir os valores dos erros preestabelecidos. Isto é feito dividindo os valores do erro individual e total (item B.1) pelo "*learn factor*".

No projeto do controlador, definido todos os parâmetros acima mencionados, o passo seguinte será definir os limites de normalização (item B.5). Em seguida, caso a janela de captura ("window") for diferente de zero, será apresentado o menu do "setup"(item B.6). Encerrado o "setup" será apresentado uma tela informando que se estará entrando no modo controle em tempo real. Nesta tela pode-se alterar o valor atual da referência. A tela seguinte é a tela do menu de opções (item B.7), a partir do que pode-se ou iniciar um treinamento (item B.8), ou iniciar a aquisição dos "true vectors"(item B.9), ou então, realizar a captura dinâmica (item B.10).

B.4 MODO I(ndirect control)

B.4.1 Introdução

O controle indireto difere do controle direto devido a necessidade de se introduzir uma rede adicional denominada rede identificadora. Esta rede permite que seja retropropagado o erro da saída do processo diretamente na

saída do controlador neural. A rede identificadora só é utilizada durante o treinamento da rede controladora após ter sido realizada uma captura dinâmica.

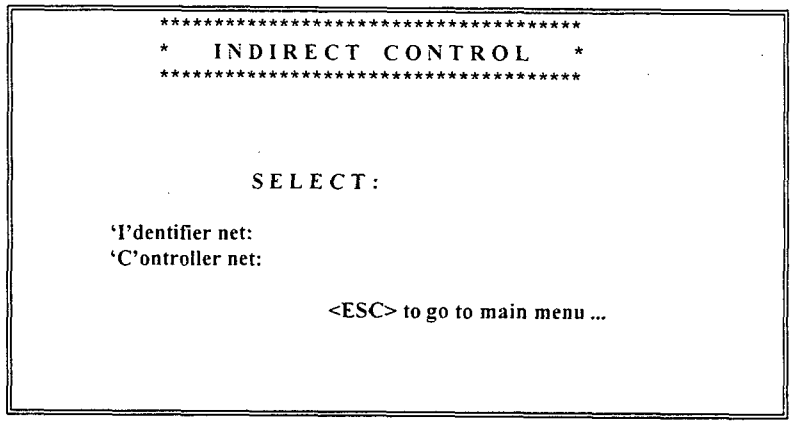


Figura 31 - Controle indireto

B.4.2 Rede identificadora

Selecionando o modo de controle indireto, inicialmente devemos proceder o treinamento da rede identificadora. Para isto selecionamos a opção "*identifier net*" (figura 31) e em seguida a opção "*setup*" (figura 32). Observe que a maioria dos parâmetros da tela do "*setup*" (figura 33) são idênticos aos dos outros modos. A primeira diferença ocorre na opção "*input ranges*". Esta opção permite que seja definido os limites, caso venha ser necessário utilizar mais de uma rede (redes especializadas) para representar o comportamento do processo. "*Input ranges*" atua de maneira similar a "reference range" das redes controladoras. Nesta opção, a seleção da ou das redes identificadoras é feita a partir do sinal que é aplicado a esta rede. Portanto, na opção "*input ranges*" devemos indicar qual são os limites de entrada ou atuação de cada uma das redes identificadoras. Para facilitar a indicação destes limites, os valores são não normalizados.

Outra diferenciação é a interpretação do número de entradas. Para as rede identificadoras temos duas opções. A primeira delas "*number of internal*

inputs" se refere as entradas da rede identificadora que estão diretamente conectadas à rede controladora. Neste caso, foi restringido o número de entradas internas a duas. Já "*number of external inputs*" se refere a outras entradas que não provenientes da rede controladora.

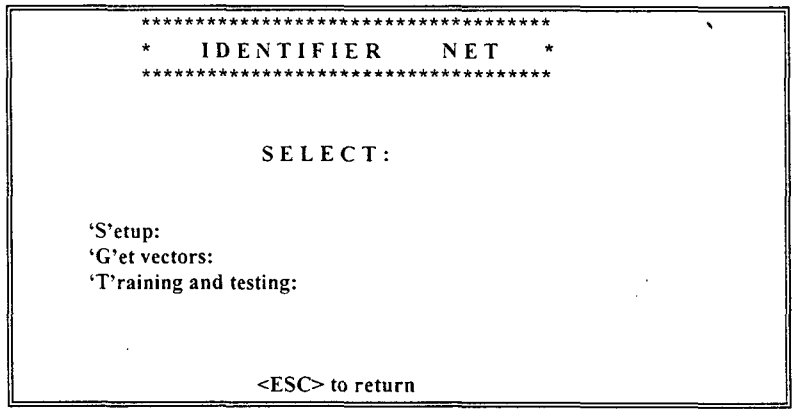


Figura 32 - Rede identificadora

É importante que o número de "delays" definidos para rede identificadora seja o mesmo que o definido para a rede controladora.

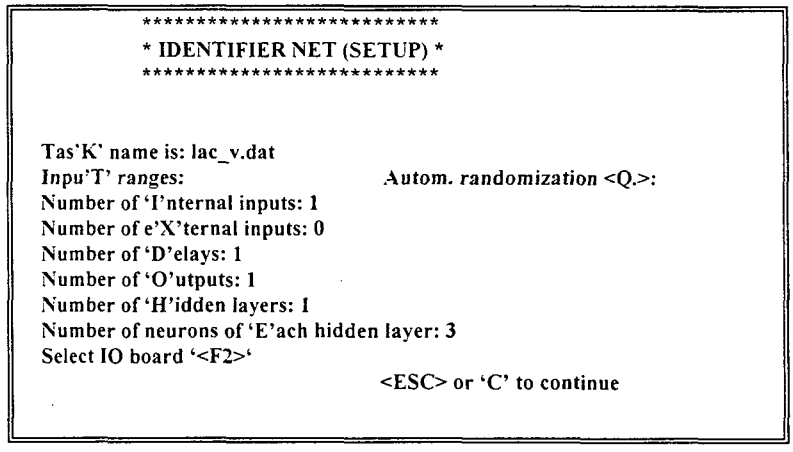


Figura 33 - "Setup" da rede identificadora

Encerrada a operação em "identifier net (setup)", voltando a tela da figura 32, devemos selecionar agora a opção "*get vectors*". Como resultado teremos então a tela da figura 34.

A opção "*window divided in blocks*?" permite que a janela de captura seja dividida ou não em blocos menores. Em cada captura pode-se alterar o tamanho do bloco seguinte, podendo variar de tamanho igual a 1 até a quantidade

restante de vetores de treinamento, de maneira a completar a janela de captura ("window"). Isto permite que sejam capturados valores estáticos (bloco de tamanho 1) ou valores dinâmicos, para o caso de blocos de tamanho maior que 1. Observe que temos duas opções "*input 1 before*" e "*input 1 after*". Estas opções correspondem exatamente aos valores que serão aplicados a entrada do processo respectivamente antes e após ter sido pressionada a tecla "C", condição que inicia a captura propriamente dita. Se o valor antes e após forem os mesmos, estaremos capturando uma relação estática entrada/saída do processo. Neste caso o tamanho do bloco deverá ser igual a 1. Se os valores antes e após forem diferentes e o tamanho do bloco for maior que 1, estaremos então realizando uma captura dinâmica. Caso tenhamos mais de uma entrada, será apresentado também os valores referentes as demais entradas.

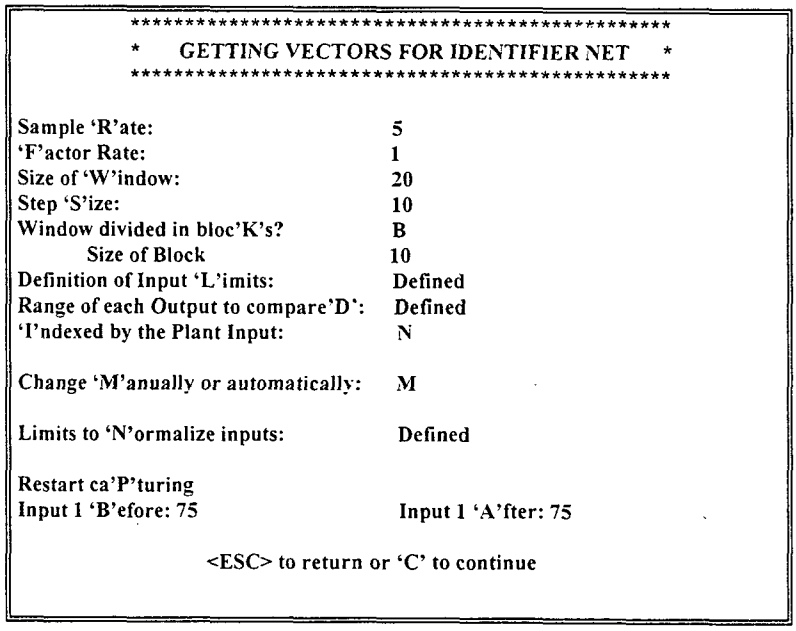


Figura 34 - Obtenção dos vetores da rede identificadora

"*Definition of input limits*" é mais um fator de segurança. Nos valores limites utilizados para a normalização, ou seja, "*limits to normalize inputs*", são indicados os limites que cada entrada da rede pode assumir. Em algumas plantas, o fato de se trabalhar próximo a estes limites em malha aberta pode ser considerado temerário. Muitas vezes o problema pode surgir devido a algum descuido o que poderia acarretar em dano ao processo. Para atenuar este problema, na opção "*definition of input limits*" pode-se definir os limites seguros de operação do processo, de modo que o próprio Ambiente se encarregará de evitar que os limites estipulados sejam ultrapassados.

"*Range of each output to compared*" funciona como um filtro durante as capturas dinâmicas. Nesta opção pode-se definir qual é o diferencial entre um valor capturado e o seguinte, evitando que desta forma, sejam capturados vetores de treinamento muito próximos ou exatamente iguais. Esta opção, em conjunto com o "*step size*" permitem que seja varrido toda a faixa de interesse para a captura dos vetores de treinamento. Por exemplo, o valor 0,1 corresponde a um diferencial de 10 %. Este valor pode ser variado para cada nova captura. O interesse de se variar este valor reside no fato de que uma variação de 10 % para valores relativamente pequenos pode não ser muito expressivo. Já para valores maiores, um diferencial de 10 % pode ser já demasiado.

"*Indexed by the plant input*" tem como objetivo fazer com que durante alguma captura dinâmica, somente sejam substituídos os vetores de treinamento que tiverem o mesmo valor de entrada dentro de uma variação de 5 %.

Esta opção foi gerada inicialmente com o intuito principal de tornar a rede identificadora também adaptativa. Com o desenvolvimento do trabalho percebeu-se que não é necessário manter-se dois elementos adaptativos em série, o que desta forma, contestamos em parte o exposto em [10]. De um outro modo, pequenas variações que porventura venha ocorrer no processo, o próprio efeito adaptativo da rede controladora será capaz de absorver. Alterações maiores que por alguma razão vierem a acontecer no processo, não podem ser consideradas como normais, e além deste aspecto, a dificuldade de se realizar aprendizados ditos on-line. Durante o tempo necessário para que ocorra um novo treinamento da rede, o controlador pode operar numa condição não segura, ainda mais se for necessário retreinar a rede identificadora também. De um modo geral, o comportamento de autoretreinamento discutido em muitos trabalhos pode ser, em tese, interessante. Já numa aplicação prática, além dos fatos já mencionados, corremos um outro risco que seria o de provocar uma alteração em todas as redes do sistema e conectá-las diretamente ao processo sem uma prévia supervisão do especialista. Cremos que esta atitude podem trazer conseqüências muito difíceis de serem previstas o que a torna não recomendável para a maioria dos processos e aplicações industriais. Em contrapartida, para contornar o problema associado àquelas situações onde o efeito adaptativo da rede controladora não venha a ser suficiente, ou até mesmo, para aquelas situações em que se deseje uma ação mais rápida, pode-se utilizar

o erro em regime permanente observado no sistema através do menu de opções (item B.7).

Caso se deseje que as entradas durante a captura dos vetores de treinamento sigam uma variação automática e senoidal, basta na opção "*change manually or automatically*", selecionar a opção automática, indicando os limites superior e inferior, o número de passos num ciclo de 360 ° e o números de pontos em cada um dos passos. O número de passos permite definir a forma da variação. Já o número de pontos por passos permite que seja definido patamares condizentes com a velocidade de resposta do processo, ou seja, permite que seja determinado quanto tempo o valor da entrada permanecerá num mesmo valor proporcionalmente à taxa de amostragem utilizada.

"*Restart capturing*" libera todos os dados anteriormente capturados, permitindo que seja reiniciado o processo de captura.

A partir deste ponto, digitando-se a letra <C> é iniciado a captura para o bloco especificado, ou então, pressionando a tecla <ESC> retornamos a tela anterior. Terminada a captura, será apresentado na tela os valores capturados, figura 35, bem como as opções para salvar estes dados, editá-los e continuar a captura.

VECTORS FOR THE IDENTIFIER NET (normalized)

	I0	I0d0	OUT
(1)	0.5906	0.5906	0.1466
(2)	0.5906	0.5906	0.1466
(3)	0.5906	0.5906	0.1466
(4)	0.5906	0.5906	0.1466
(5)	0.5906	0.5906	0.1514
(6)	0.5906	0.5906	0.1514
(7)	0.5906	0.5906	0.1514
(8)	0.5906	0.5906	0.1514
(9)	0.5906	0.5906	0.1514
(10)	0.5906	0.5906	0.1563

'S'ave e'D'itor 'R'epeat 'N'ext <ESC> to return

Figura 35 - Captura de vetores da rede identificadora

Mantendo a seqüência referente a rede identificadora, e retornando a tela da figura 32, devemos selecionar a opção "*training and testing*". Em seguida, para iniciarmos o treinamento da rede identificadora devemos selecionar a opção

"*off line training*" da tela figura 36. Feito isto, será apresentada a tela da figura 37.

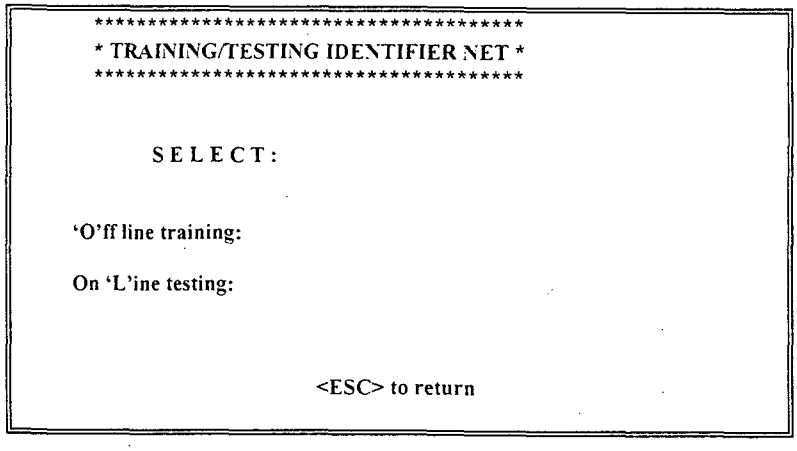


Figura 36 - Treinamento e teste da rede identificadora

Em "*input data file*", devemos indicar o nome do arquivo de dados que contém os vetores de treinamento que serão utilizados para o treinamento da rede identificadora.

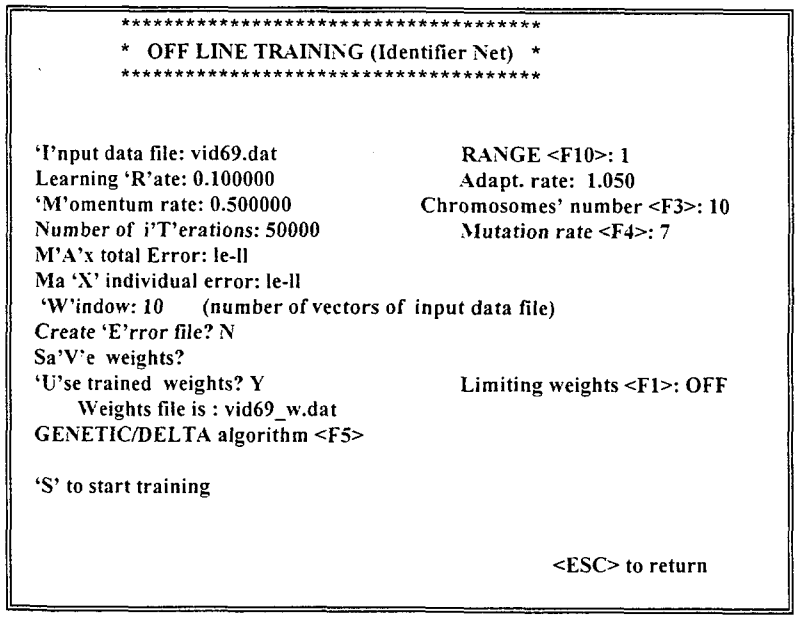


Figura 37 - Treinamento da rede identificadora

"*Create error file*" permite que seja criado um arquivo de dados de nome "crit_id.dat" que conterá todas as variações do valor do erro total durante o treinamento da rede identificadora.

Pressionando-se a letra <S> será iniciado o treinamento propriamente dito da rede identificadora de acordo com o exposto no item B.8.

Após concluído o treinamento pode-se, de maneira on-line, comparar o desempenho em malha aberta do processo e da rede neural identificadora treinada. Para isto, basta voltarmos a tela da figura 36 e selecionar agora a opção "*on line testing*".

Como resultado, teremos a tela da figura 38. Os valores das entradas podem ser alterados diretamente digitando o número da entrada, no caso "0" e em seguida o novo valor, ou se tivermos mais de uma entrada, "1" para a segunda entrada e o valor. Também é possível provocar uma variação contínua para a entrada "0" usando as teclas de <+> e <-> ou, <*> e </> para a entrada "1". Estas teclas provocam o incremento/decremento automático dos valores aplicados diretamente a rede identificadora e a planta.

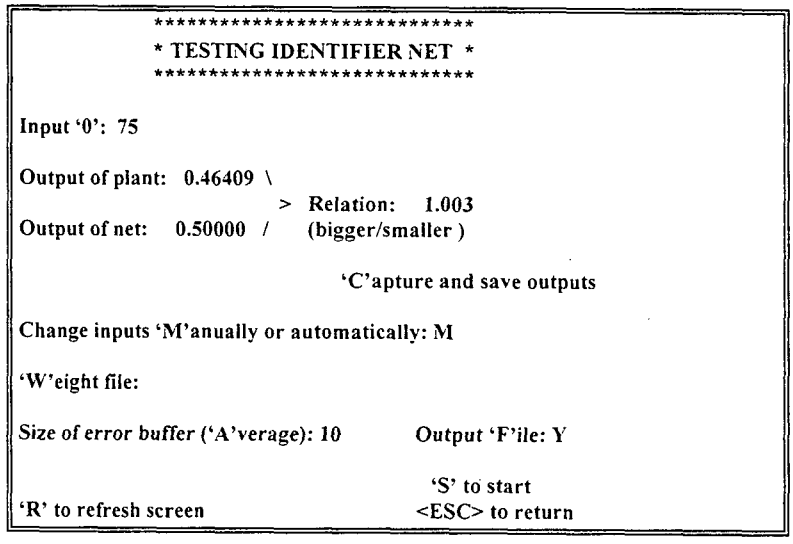


Figura 38 - Testando a rede identificadora

O valor da relação entre as duas saídas pode ser lido diretamente. Para evitar oscilações na leitura do valor do erro, é apresentado como resultado a média aritmética dos últimos valores através de um memória circular. A dimensão desta memória em número de valores armazenados, é dado por "*size of error buffer (average)*".

Caso se deseje que as entradas sigam uma variação automática e senoidal, basta na opção "*change inputs manually or automatically*" indicar os limites superior e inferior, bem o número de passos num ciclo de 360 ° e o números de pontos lidos em cada um dos passos.

Também é possível salvar os valores lidos diretamente num arquivo de dados através da opção "*output files*". Quando acionada esta opção, aparecerá uma mensagem adicional na tela "capture and save outputs". Toda vez que for pressionada a tecla <C> será salvo no arquivo "out_id.dat" o valor correspondente da saída da planta ou processo, da saída da rede identificadora e o erro médio.

Deve-se observar que o teste comparativo só se iniciará quando pressionada a tecla <S>, e de mesma forma, só terminará quando novamente for pressionada a mesma tecla ou então a tecla <ESC>.

B.4.3 Rede controladora

Para iniciar o treinamento da rede controladora devemos voltar a tela indicada na figura 31 e selecionarmos a opção "*controller net*". Como resultado teremos o apresentado na tela da figura 39.

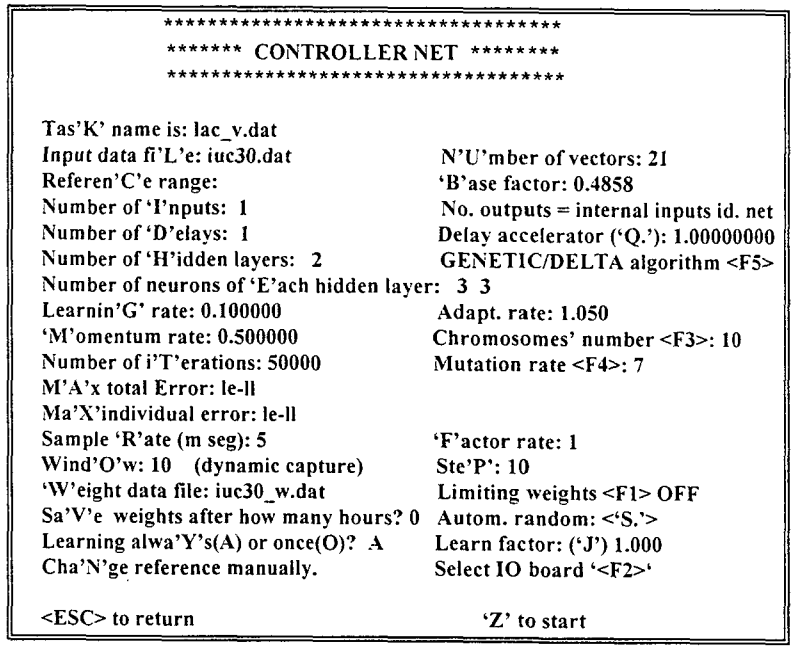


Figura 39 - Rede controladora (controle indireto)

Os procedimentos de definição da rede controladora do modo de controle indireto e direto se diferenciam em alguns poucos itens. No que tange a tela principal, os itens a serem preenchidos são exatamente os mesmos.

Na rede controladora do controle indireto o número de neurônios de saída desta rede é definido pelo número de neurônios de entrada da rede identificadora. Este parâmetro só pode ser alterado no "setup" da rede identificadora.

No item B.6 que trata do "setup" do modo de captura dinâmica, no caso do controle direto deverá ser definido o tipo do ganho (linear ou exponencial) e o valor deste. No controle indireto não existe a necessidade destes ganhos.

Outra diferença entre o controle indireto e do controle direto, é que no caso do controle indireto, nos vetores de treinamento utilizados para o treinamento da rede controladora, aparecerá uma ou mais colunas que correspondem aos atrasos ("delays") da ou das saídas.

B.5 Definição dos Limites das Entradas e Saídas

Após cada tela principal dos vários modos e também em algumas opções específicas, é necessário definir os limites de cada entrada/saída. Estes valores são usados para normalização dos dados de entrada e saída das redes neurais.

A tela correspondente é acionada somente quando os dados referentes a topologia da rede estão todos definidos.

Inicialmente é solicitado os limites mínimo e máximo da referência, em seguida das entradas e por último, das saídas.

B.6 "SETUP" (Controle Direto e Indireto)

Durante a realização dos controladores neurais, tanto no modo direto como no indireto, devemos definir os aspectos gerais com respeito a captura dinâmica.

Esta opção é acionada caso a janela de aquisição de vetores de treinamento ("window") for diferente de zero.

Inicialmente é apresentado a tela da figura 40 onde temos os dados predefinidos. Caso deseje-se alterar algum item em específico é necessário responder com <N> na linha "confirm (Y) <CR> or (N)?".
Durante a alteração, todos os itens correspondente ao "setup" serão acionados. Caso se queira manter algum item inalterado, basta teclar <ENTER> que será mantida a condição anterior.

A primeira opção se refere a *partição da janela de captura.* "All vector" pressupõe que a janela será única, sem divisões. Caso seja necessário dividí-la em porções menores, digitar <B> de "blocks". Será apresentado em seguida o tamanho total da janela e em seguida deve ser preenchido o tamanho de cada bloco contado em número de vetores ou linhas. Se o tamanho do bloco não for múltiplo do tamanho da janela, então o último bloco terá tamanho inferior aos demais.

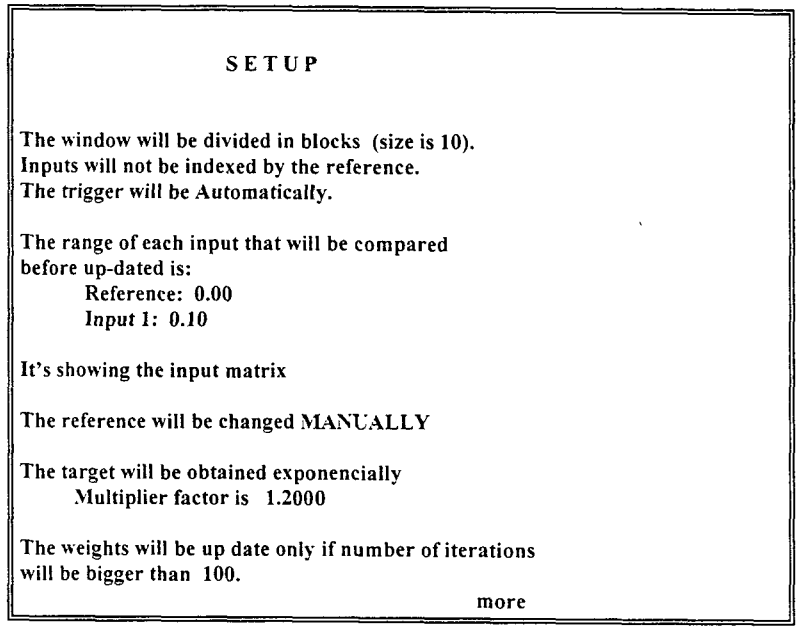


Figura 40 - Tela "setup" para captura dinâmica

A mensagem seguinte indicará se as atualizações dos vetores de treinamento adquiridos serão *indexadas pela referência ou não.* Caso a opção seja <N>, teremos que as novas capturas serão feitas com sobreposição de todos os valores anteriormente capturados, ou seja, os valores antigos, se houverem, serão todos substituídos. Já se a opção for <Y>, deveremos definir o percentual de comparação. Neste caso, serão substituídos somente os vetores de treinamento que tiverem o mesmo valor da referência, dentro da margem ou percentual indicado. Isto permite, que durante o treinamento adaptativo, não haja um sobretreinamento de um único ponto da região de controle, evitando que, após contínuas capturas, ocorra o "esquecimento" das demais regiões. Desta forma, será sempre mantida a distribuição dos vetores de treinamento de acordo como foram inicialmente definidos.

O passo seguinte é determinar se o *preenchimento da matriz de vetores de treinamento* ("input matrix") será feita de maneira automática ou através de algum *disparo externo* ("external trigger"). Preenchimento *automático* é aquele que encerrado um treinamento, a captura de novos vetores de treinamento se dará de forma automática. Já através de *disparo externo*, uma nova captura só iniciará quando houver uma variação em uma das entradas além de um limite preestabelecido. Caso seja optado por preenchimento através de disparo externo, devemos inicialmente selecionar qual das entradas servirá como disparadora. Em seguida deveremos indicar qual é a máxima variação aceitável, a partir do que, será iniciada a captura.

A próxima questão se refere a *variação do sinal de referência durante a captura*. Neste item temos três possibilidades: na primeira é manter a variação controlada *manualmente*. Para este caso, devemos indicar o valor da tensão de referência antes e após o disparo. Esta condição é a ideal para ensaios do tipo degrau. Na segunda opção temos a referência variando de maneira *automática* seguindo uma variação senoidal. É necessário indicar qual o valor mínimo e máximo permitidos para a referência, quantos pontos ou valores diferentes para uma excursão de 360 °, e também, quanto tempo deverá a referência ser mantida num mesmo valor. Isto é tomado em relação a taxa de amostragem utilizada. A terceira opção permite que sejam definidos *valores quaisquer* para a referência. Caso já tenha definido alguns valores, estes valores serão apresentados de início, com a opção de alterar um ou todos. Para cada valor de referência definido deve-se indicar também qual o tempo em número de taxas de amostragem, que a referência permanecerá no mesmo valor. Isto permite que sejam estabelecidos patamares de acordo com a dinâmica da planta.

A opção seguinte indica se a *matriz dos vetores capturados será ou não mostrada* logo após encerrado a captura. Se estivermos utilizando controle adaptativo por retreinamento disparado automaticamente, deveremos inibir a apresentação da "input matrix", ou seja, da matriz onde são salvos todos os vetores de treinamento capturados dinamicamente. Do contrário, toda vez que for realizado uma nova captura, a matriz de vetores de treinamento será mantida na tela até que seja pressionado alguma tecla. Isto não permitirá que após uma captura haja o disparo automático do treinamento. Para as demais situações, é interessante mostrar a matriz dos vetores a cada captura.

Somente para quando estamos utilizando o *controle direto*, devemos definir qual é o tipo de *ganho* que iremos utilizar, se será *linear ou exponencial*, e também o seu respectivo valor. Esta opção não será apresentada quando estivermos trabalhando com o controle indireto.

Como última opção desta tela temos *o número mínimo de iterações durante o treinamento a partir do que será feita a atualização dos pesos da rede controladora em tempo real*. Esta opção leva em consideração que durante um treinamento, se a partir de algumas poucas iterações o treinamento encerrar, isto pode caracterizar que praticamente não houve aprendizado ou mudança significativa que justifique a alteração dos pesos da rede controladora em tempo real, não sendo portanto copiado os pesos da rede que está sendo treinada para a rede que está controlando efetivamente a planta.

B.7 Menu de Opções

Logo no início da tela da figura 41 temos a indicação do valor da referência. Digitando-se a letra <R> será alterado o valor da referência imediatamente. Já se for pressionada a tecla <A>, o valor correspondente da referência só será alterado quando for disparado uma nova captura. Isto é feito através das teclas <N> de "*next*" e/ou <R> de "*repeat*". "*Next*" indica que os novos valores capturados serão armazenados no bloco seguinte, caso haja mais de um bloco. Já "*repeat*" significa que os novos vetores de treinamento capturados serão sobrescritos no bloco atual. Deste conjunto de opções é possível aplicar ao processo qualquer sinal tipo degrau, sendo provavelmente a melhor forma de se capturar as características dinâmicas do processo.

"*(+) velocity(-)*" permite variar a velocidade de resposta da rede controladora acoplando a esta um elemento de comportamento similar ao de um diferenciador nos controladores PIDs clássicos. Esta variação de velocidade é feita após a rede ter sido treinada, diretamente sobre a rede em tempo real. O efeito buscado aqui é a alteração na velocidade de resposta do controlador de modo que possa ser analisado o comportamento da rede neural em tempo real.

A qualquer instante, pressionando a tecla <*>, será imediatamente desativada esta opção.

"*Steady state error*" permite criar um mecanismo de contorno caso o erro de regime permanente ultrapasse a valores predefinidos. É possível definir faixas de atuação e ainda, para cada faixa escolher uma entre duas opções: A primeira mantém a saída da rede num valor fixo caso tenha-se atingido o limite estabelecido para esta faixa. A segunda, atingido o limite da faixa, comuta a rede em tempo real por outra previamente construída. Quando se entra em "*steady state error*" é perguntado se deseja continuar, desativar a função ou retornar a tela anterior. Continuando, é perguntado quantos valores serão lidos de modo a obter a média aritmética do valor do erro em regime permanente. O erro é determinado em função da média aritmética destes valores. Em seguida, devemos indicar quantas faixas queremos e como será a atuação dentro da faixa, se por fixação do valor da saída ou pela troca da rede controladora em tempo real. Nestas duas opções, devemos indicar também qual é o percentual de variação permitido. Se for escolhido por fixação da saída, devemos entrar com o valor da saída (número não normalizado), ou se por troca, qual é o nome do arquivo de dados de pesos da nova rede.

Após ter sido selecionado a opção de avaliação instantânea do erro de regime permanente, e havendo alguma perturbação no sistema de maneira a atingir o limite de alguma das faixas predefinidas, ocorrerá que, no caso que a atuação prevista para a faixa tenha sido a troca de rede, será somente indicado que ocorreu a troca e o controle continuará normalmente. Já, se a opção definida foi a fixação do valor da saída, será então, imediatamente congelado o valor da saída do controlador no valor escolhido, e em seguida encerrado o controle. Havendo atuação, pressionando-se a tecla <ESC> será desativada a opção de regime permanente.

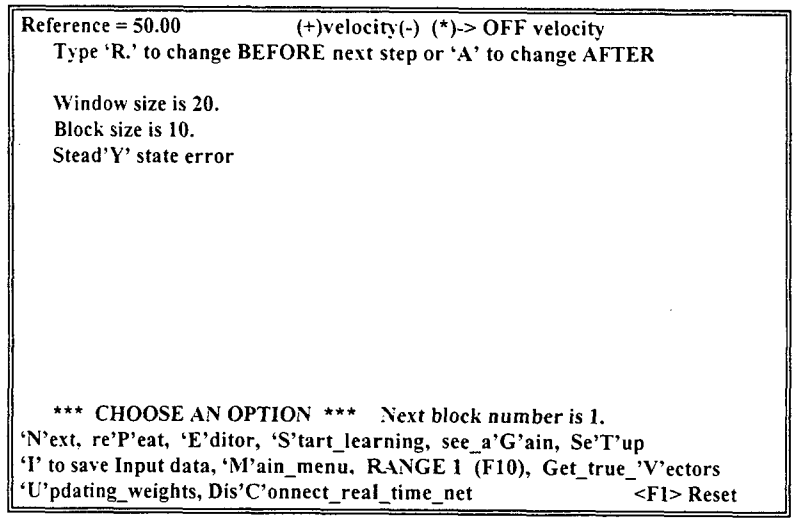


Figura 41 - Menu tempo real

"*Editor*" permite que seja alterado diretamente os vetores capturados, bastando indicar a coluna e a linha do valor a ser alterado.

"*Start learning*" dispara o processo de treinamento de acordo com o item B.8.

"*See again*" permite que seja mostrada na tela todos os vetores capturados.

"*Setup*" aciona a opção descrita no item B.6.

"*I to save input data*" salva todos os vetores capturados para utilização futura.

"*Main menu*" acessa a tela inicial do respectivo modo, permitindo que seja redefinidos alguns parâmetros. Devemos atentar que parâmetros que alterem a topologia das redes, tais como número de entradas, número de camadas e outros, não podem ser alterados neste instante, pois isto poderia provocar uma inconsistência com a rede que está trabalhando em tempo real. O próprio Ambiente bloqueará qualquer tentativa neste sentido. Caso seja necessário a alteração da topologia da rede, deve-se pressionara a tecla <ESC> retornando a tela da figura 27 e selecionar novamente o modo correspondente. Isto fará com que seja desinstalado a rede em tempo real.

"*Range n*" está associado diretamente ao número de redes especializadas ou paralelas que foram definidas. Indica qual rede que será treinada de acordo com a seqüência gerada durante a opção "reference ranges" nos menus das redes

controladoras dos modos de controle direto e de controle indireto. Observe que os vetores de treinamento são sempre os mesmos independentemente de qual faixa estamos selecionando. A discriminação de qual vetor da matriz de vetores de treinamento será utilizado em cada faixa é feita de maneira automática durante a fase de treinamento propriamente dita.

"*Get true vectors*" segue o exposto no item B.9. Basicamente, nesta opção são encontradas todas as facilidades para captura e utilização dos vetores de treinamento obtidos em malha aberta diretamente do processo, vetores estes que serão utilizados na primeira fase de treinamento da rede controladora, tanto para o controle direto como no indireto.

"*Updating weights*" permite que logo após ter-se encerrado algum treinamento, que a rede recém treinada seja utilizada para atualizar os pesos da rede que está em tempo real. Esta atualização será automática caso o treinamento ocorrer com sucesso.

"*Disconnect real time net*" funciona como uma chave liga/desliga da rede controladora em tempo real. Quando não for importante manter ativa a rede em tempo real, a desconexão desta, permite que seja concentrado o tempo de processamento no treinamento, diminuindo por conseguinte, o tempo final do treinamento.

"*Reset*" reinicializa todos os indicadores utilizados durante a captura dinâmica.

B.8 Treinamento

Após obtido os vetores de treinamento, podemos iniciar o treinamento propriamente dito.

Nas telas principais de cada um dos modos do Ambiente é possível selecionar qual dos algoritmos de treinamento serão utilizados. Basicamente, são

oferecidos dois algoritmos, que são: algoritmo genético e o baseado num variante da regra delta, conhecido como "backpropagation". Sobre o "backpropagation" também é possível utilizar em conjunto uma opção que opera com características inspiradas no método de "simulated annealing".

Como conseqüência, o algoritmo genético é mais recomendado para as primeiras etapas de treinamento, pois proporciona uma busca mais global. O algoritmo baseado na regra delta apresenta um bom desempenho na busca de soluções locais, ou seja, é mais adequado para a etapa final de treinamento, onde se busca atingir o mínimo da superfície do erro de uma rede neural multicamadas. "Simulated annealing" busca melhorar a característica de busca global do "backpropagation". Devemos salientar, que nenhum método de treinamento existente até hoje garante a busca da solução ótima, principalmente caso a superfície do erro seja muito convoluída, ou seja, complexa e multimodal. No entanto, usando-se uma solução híbrida pode-se chegar a resultados bastante satisfatórios. Caso seja selecionado algoritmo genético e o "backpropagation", sempre será executado em primeiro o algoritmo genético e por conseguinte, o "backpropagation". Durante o treinamento adaptativo é recomendado somente utilizar o algoritmo "backpropagation", visto que, as alterações que se busca trabalhar são sempre próximas ao ponto de operação, não necessitando de grandes buscas.

Somente durante o treinamento no modo de controle direto, será apresentado na tela qual *tipo de ganho* que está sendo utilizado, sendo possível alterar além do valor, o próprio tipo de ganho.

Em "*range n*" é indicado qual é a rede que está sendo treinada caso tenhamos redes especializadas ou paralelas.

B.8.1 Treinamento Genético

Em sido selecionado a *algoritmo genético*, será apresentado a tela da figura 42. Durante a fase de treinamento é permitido alterar a "*mutation rate*" ou taxa/número de mutações realizadas a cada iteração. Este fator determina

quantas mutações serão realizadas a cada ciclo de treinamento. Em experimentações, observou-se que este número deve ser inferior a cerca de 70 % do número de cromossomos definidos. Para número de mutações superiores a este, é interessante utilizar os mecanismos de controle de divergência apresentados a frente.

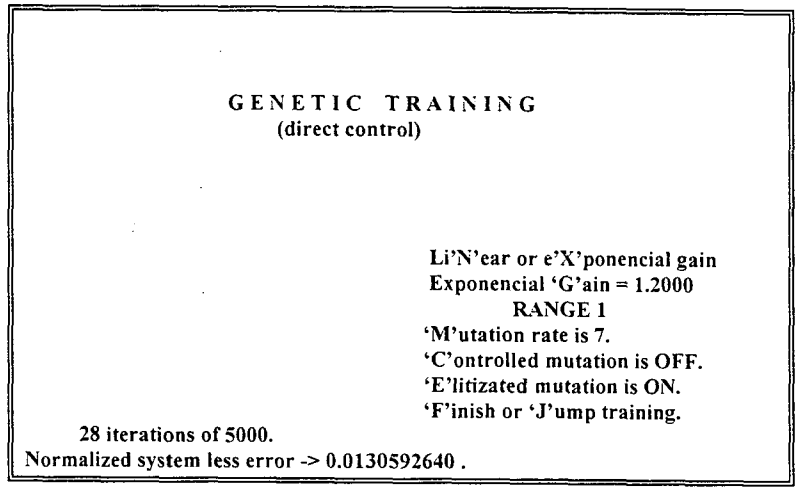


Figura 42 - Treinamento genético

Um dos grandes problemas enfrentados durante a utilização de algoritmos genéticos é a dificuldade de manter a convergência da função objetivo. O que se busca é evitar a perda da informação genética, sempre permitindo que os indivíduos mais adaptados tenham conseqüentemente mais chance de sobrevivência. No Ambiente aqui apresentado duas opções foram implementadas neste sentido que são "*controlled mutation*" e "*elitazed mutation*". Ambas operam como liga/desliga e facilitam em muito o trabalho do usuário, evitando que este tenha que ficar acompanhando o comportamento da função objetivo, atento ao instante em que a convergência para de acontecer.

"*Controlled mutation*" a cada nova mutação testa se o indivíduo mutado teve uma melhora na sua adequabilidade ("fitness") ou não. Caso tenha havido alguma melhora, a mutação será considerada com sucesso e por sua vez é mantida. Caso contrário, a mutação não será aceita e será mantido o alelo correspondente ao instante anterior a mutação. Com a intenção de reduzir o problema da convergência prematura, este critério de substituição é relaxado com uma probabilidade de 0,1. De uma certa forma, isto faz com que um indivíduo não seja definidamente eliminado do espaço de busca e também evita

que este espaço seja comprimido de maneira radical podendo comprometer a busca da solução global.

"*Elitizated mutation*" evita que o material genético dos melhores indivíduos seja perdido durante o processo de mutação. Neste método, a cada ciclo, só é permitida a mutação sobre os 75 % piores indivíduos, preservando-se integralmente os melhores desta população. Este critério necessita conseqüentemente menos processamento computacional que o anterior, o que resulta num tempo de treinamento final menor que o utilizando na "controlled mutation".

"*Finish*" ou <ESC> implica no encerramento total do treinamento e "*jump*" conclui o treinamento genético. Caso o treinamento baseado na regra delta tenha também sido selecionado, "*jump*" fará então que seja iniciado imediatamente este outro algoritmo. Caso não, será apresentado o resultado do treinamento de acordo com o item B.8.3.

B.8.2 Treinamento "Backpropagation"

Neste método (figura 43), é necessário definir dois parâmetros que são "*learning rate*" ou taxa de aprendizagem e "*momentum rate*" ou taxa de momentum. Para se atingir o mínimo de fato de uma função, o valor do "*learning rate*" deve ser o menor possível. Em conseqüência, quanto menor for este valor, tanto maior será o tempo de treinamento. Isto poderia então cair numa solução de compromisso. Para contornar este problema, o ajuste do "*learning rate*" é feito de maneira automática. Inicialmente podemos definir um valor mais alto para o "*learning rate*" e toda vez que ocorre um processo de divergência da função objetivo, o "*learning rate*" será então divido pelo valor do "adapt. rate" definido na tela principal de cada modo. Esta situação, provocará uma redução gradativa e automática do valor do "*learning rate*", condição que se apresenta muito interessante para este tipo de algoritmo. Durante os ensaios, verificou-se que o valor do "*learning rate*" deve ser inferior a 3. Caso estejamos utilizando somente o algoritmo "backpropagation", recomenda-se manter o valor inicial entre 1 e 3.

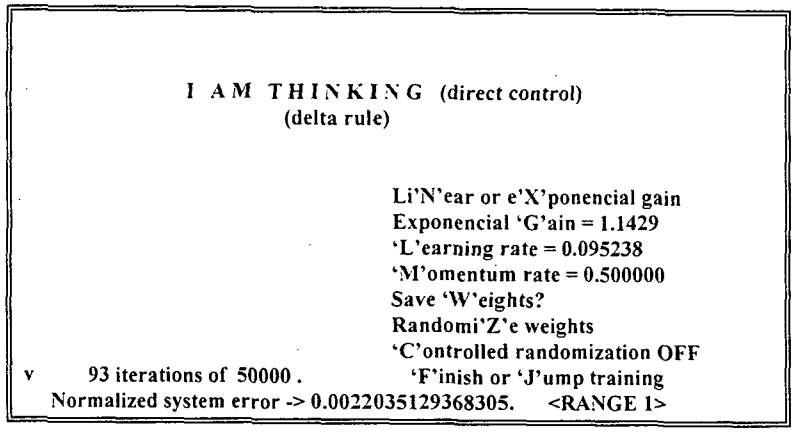


Figura 43 - Treinamento "backpropagation"

Já, caso tenha sido selecionado também o algoritmo genético de treinamento, recomenda-se que o "*learning rate*" seja inferior a 0,1.

Mesmo critério de ajuste automático é efetuado sobre os ganhos do controle direto.

Quanto ao "*momentum rate*", seu valor sempre deve ser inferior a 1. Valores usuais são tipicamente entre 0,5 e 0,7. Outros valores podem ser utilizados. Não se recomenda grandes variações deste parâmetro durante o processo de treinamento, podendo isto acarretar um encerramento abrupto do treinamento, sem no entanto atingir o mínimo da função da região de busca.

Através da opção "*save weights*" é permitido a qualquer instante salvar em um arquivo de dados os pesos da rede que está sendo treinada.

"*Randomize weights*" realiza uma randomização sobre todos os pesos da rede.

"*Controlled randomization*" é do tipo liga/desliga. Esta opção realiza um trabalho não idêntico, mais inspirado no algoritmo de "simulated annealing". O problema principal no algoritmo de "simulated annealing" é a definição das temperaturas de cada tentativa. Isto, sem dúvida, implica na necessidade que haja um relativo domínio do processo analisado, sem contar na necessidade de definição de mais parâmetros por parte do usuário. A solução adotada foi a de se utilizar um gerador de números randômicos que possuísse um ganho associado, sendo que este ganho é proporcional a distância euclidiana do erro

atual com o erro desejado. Os parâmetros são definidos na tela principal de cada modo como "autom. random.", bastando simplesmente definir o número máximo de tentativas e o valor mínimo do "learning rate" que disparar um novo processo de randomização. Nenhum destes fatores é crítico para o treinamento, sendo que pode ser utilizado os valores predefinidos.

"Finish" ou <ESC> encerra ou aborta o treinamento e "jump" mostra o resultado do treinamento até aquele instante.

B.8.3 Apresentação do Resultado do Treinamento.

Basicamente um treinamento pode encerrar com sucesso ou não, ou ainda, por intervenção do usuário através das opções de "finish" e "jump". Salvo a opção "finish", as demais será apresentado na tela a descrição completa de todos os pesos resultantes do treinamento realizado. Treinamento encerrado com sucesso é aquele onde foi possível alcançar o valor do erro desejado. Neste caso, nos modos de controle, os pesos da rede treinada serão copiados na rede controladora em tempo real. Treinamento sem sucesso é aquele que é encerrado devido ter-se atingido o número limite de iterações definido. Em ambos os casos será informado a condição de encerramento na tela.

Além desta mensagem e da descrição dos pesos, é apresentado o resultado comparativo do treinamento em função dos vetores utilizados durante o treinamento. É apresentado o valor alvo ou desejado e o respectivo resultado da rede recém treinada, além do erro global normalizado. Devemos observar, que havendo mais de uma rede selecionando um mesmo conjunto de treinamento (caso das redes especializadas ou paralelas), será somente apresentado o resultado do teste feito exclusivamente com os vetores de treinamento realmente utilizados no treinamento da rede em questão, ou seja, serão apresentados os resultados dos vetores que estiverem dentro da faixa ("range") da rede selecionada.

B.9 Obtenção dos "TRUE VECTORS"

Do menu de opções, selecionando a opção "*get true vectors*", será apresentado a tela da figura 44.

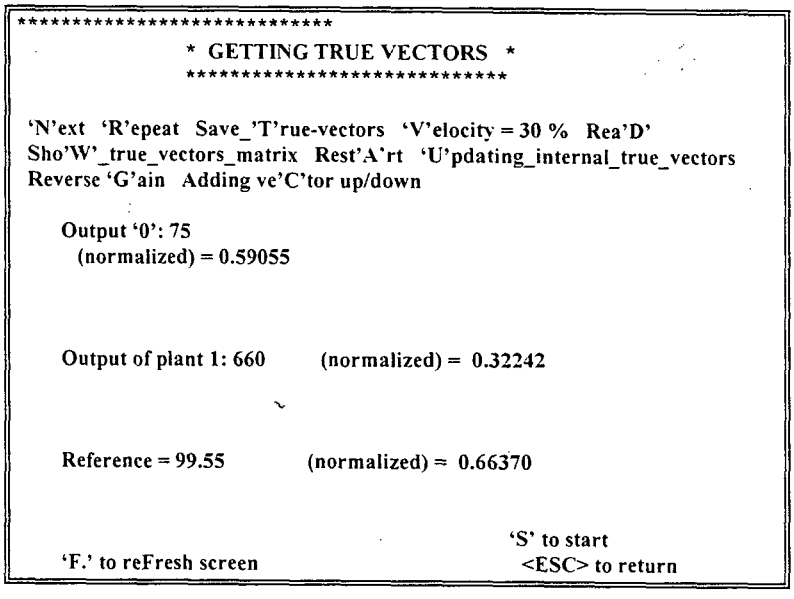


Figura 44 - Obtenção dos "true vectors"

A captura dos "true vectors" é feita mediante a variação do sinal que é aplicado diretamente a entrada do processo, indicado como "*output*". O controle de variação pode ser feito digitando o valor correspondente após teclar-se o número da saída escolhida. Outra maneira de variar o sinal a ser aplicado na entrada do processo é utilizando-se as teclas <+> e <-> para "*output 0*" e, <*> e </> para "*output 1*" que por sua vez incrementam/decrementam automaticamente o valor deste sinal. Observar que foi limitado o número máximo de saídas a duas.

As capturas são feitas ponto a ponto, sendo importante aguardar que o sistema esteja perfeitamente estável antes de realizar a captura. Recomenda-se que sejam adquiridos vetores tanto no sentido crescente da referência como decrescente. A escolha dos pontos devem ser feita de acordo com as não linearidades do processo analisado e o número de redes paralelas que se está utilizando.

Após definidos os pontos de captura, devemos também definir qual a velocidade ou o ganho do controlador. Isto é feito através da opção "*velocity*".

Em seguida devemos definir se o controle terá ação direta ou reversa ("*reverse gain ou direct gain*"). No capítulo 5, item 5, sobre pré-treinamento são dadas maiores informações. Tipicamente os controladores apresentam ação reversa. Ação reversa implica que quando aumenta o sinal realimentado a partir da saída do processo, o controlador por sua vez diminui o sinal aplicado na entrada do processo, restabelecendo o equilíbrio.

A opção "*adding vector*" tem quatro alternativas. Pressionando a tecla <C> pode ser escolhido se será acrescentado um vetor acima, ou então um vetor abaixo, ou ainda um a acima e outro abaixo ou mesmo nenhum. Ver figura 12. A introdução destes vetores está associado diretamente ao efeito do somador tipicamente encontrado nos controladores PIDs, que resulta no erro efetivo que será então aplicado ao controlador, como mostrado na figura 30. Também o efeito da velocidade acima comentado tem atuação diretamente sobre estes vetores que são adicionados de uma forma "off-line". Desta maneira, para cada vetor adquirido do processo, poderão ser acrescentados até mais dois outros.

A captura propriamente dita deve ser feita diretamente pelas opções "*next*" e "*repeat*". "*Next*" acrescenta mais um vetor aos já capturados. Já "*repeat*" sobrescreve o último vetor capturado. "*Restart*" reinicia a captura, anulando as capturas anteriormente feitas.

"*Save true vectors*" salva os vetores capturados num arquivo de dados.

"*Read*" permite que seja carregado um conjunto de vetores previamente gerados e continuar a inserir outros vetores a partir do último anteriormente lido.

"*Show true vectors matrix*" mostra a qualquer instante os vetores já capturados.

"*Updating internal true vectors*" copia os vetores capturados diretamente na matriz de vetores que serão utilizados no treinamento.

A captura propriamente dita se inicia quando pressionado a tecla <S> de "*start*". De mesma forma, pressionando-se novamente a tecla <S> de "*stop*" é cessada captura. Para continuar a captura basta outra vez pressionar a tecla <S>.

B.10 Captura Dinâmica

A captura dinâmica é feita diretamente sobre a tela da figura 41 do item B.7. O item B.6 referente ao "setup" permite que seja definido todos os parâmetros gerais da forma de realização desta captura.

A captura dinâmica é útil quando da fase de projeto do controlador neural, permitindo que seja incorporado ao controlador as características dinâmicas do processo. Também é aplicável durante o treinamento adaptativo, onde as capturas são disparadas automaticamente pelo próprio Ambiente.

Durante a captura dinâmica de novos vetores de treinamento é introduzido um sinal "-" antes dos valores capturados como saída do controlador. Isto é necessário para que haja diferenciação dos valores capturados como "true vectors" dos valores capturados dinamicamente. No treinamento supervisionado é necessário que seja indicado qual deve ser o valor alvo ou de saída da rede em treinamento. No caso específico dos "true vectors", o valor alvo é o próprio valor que se está injetando no processo. Já, para a captura dinâmica, não temos diretamente este valor. Conseqüentemente, o valor alvo é obtido de uma forma indireta. No caso do controle indireto, é usado a rede identificadora para retropropagar o erro da saída do processo para a saída do controlador. Para o controle direto, isto é feito a partir de uma expressão matemática, onde são definidos ganhos que são aplicados sobre o próprio valor corrente da saída da rede controladora. Para o caso da captura dinâmica, a visualização dos valores de saída da rede é importante somente para que o projetista possa observar como está o comportamento da rede que está sendo treinada, e verificar, se foi definido de uma forma adequada os filtros de captura (item 12.6 - "setup") e o intervalo entre capturas ("step" - menu principal dos modos de controle).

Esta informação também é utilizada para que, durante o treinamento propriamente dito, o Ambiente possa discriminar quando é ou não um "true vector", já que o processo de treinamento é diferenciado para os "true vectors" e para os vetores capturados dinamicamente. Isto permite também que mesmo após os "true vectors" terem sido armazenados num mesmo arquivo de dados contendo vetores que foram capturados dinamicamente, os algoritmos de treinamento possam diferenciá-los.

B.11 Compilação, Ambiente de Programação e Hardware

Para realização deste trabalho foi utilizado o ambiente para linguagem procedural C da Borland 3.0. As principais seleções do ambiente Borland são as seguintes:

- Linker output - Standard DOS EXE
- Prolog/Epilog - DOS standard
- Model - Huge
- Floating point - 80287/387
- Instruction Set - 80386
    - fast floating point
    - fast huge pointers
- Optimization Options:
    - Global register allocation
    - Invariante code motion
    - Induction variables
    - Loop optimization
    - Suppress redudant loads
    - Copy propagation
    - Dead code elimination
    - Jump optimization
    - Inline intrinsic functions

Com o intuito de facilitar a organização do programa fonte como um todo, foi dividido em módulos como segue:

1. DECL_10.C: Contém a inicialização de todas as variáveis utilizadas nos vários módulos.
2. CONNE_10.C e CONTR_10.C: Contém as rotinas referentes a rede controladora do modo controle indireto.
3. IDENT_10.C: Contém as rotinas referentes a rede identificadora do modo de controle indireto.
4. DIREC_10.C e DIRMI_10.C: Contém as rotinas da rede controladora do modo de controle direto. DIRMI_10.C contém também a parte referente a análise de consistência dos dados.
5. MIS_10.C: Contém as rotinas do modo "learning" e "output generation", além de outras rotinas referentes a alocação dinâmicas dos ponteiros, leitura e gravação em arquivos de dados, entre outras de propósito geral.
6. NEURAL10.C: Contém as rotinas de alteração do relógio interno do microcomputador, rotinas de instalação das redes em tempo real, testes/filtros para captura dinâmica e a rotina principal (main).
7. IO_BOARD.C: Contém as rotinas referente a placa de aquisição de dados A/D.
8. GENE_10.C: Contém as rotinas referente ao treinamento genético.
9. Headers:
    - DECL_10.H - Contém a definição das variáveis utilizadas em todos os módulos e a definição das constantes.
    - NETI.H - Contém as bibliotecas que deverão ser incluídas nos vários módulos.

Para completar, é necessário manter também o arquivo que contém as bibliotecas de acesso da placa A/D. O arquivo utilizado é o AX5412CL.LIB. Este arquivo deve ser selecionado de acordo com o modelo de memória utilizado, que neste caso é o "huge".

Nos módulos CONNE_10.C, DIREC_10.C, IDENT_10.C, MIS_10.C e NEURAL10.C, na última rotina destes módulos tem-se a rotina principal do módulo que determina o seqüenciamento das demais rotinas do respectivo módulo.

Com relação ao hardware periférico utilizado, como pode ser visto na figura 20, da saída do gerador foi acoplado um transformador isolador de relação 150 V / 5 V, no caso de ser utilizado o transdutor CA/CC, ou então, um transformador 150 V / 2,5 V para o caso de ser utilizado o filtro FIR (Finite Impulse Response), filtro este que é implementado em software. Para o filtro FIR, a tensão alternada do secundário do transformador é aplicada diretamente ao conversor A/D.

Para manter compatibilidade com a maioria dos processos industriais, foi optado por utilizar o transdutor CA/CC, que neste caso apresenta uma relação de 10 V_{CA} / 5 V_{CC}.

Nos ensaios realizados, a ponte de tiristores é comandada digitalmente através de um circuito eletrônico desenvolvido especialmente para este projeto, circuito este que é excitado pela saída paralela do microcomputador. Foi optado em se utilizar controle de disparo digital em vez de analógico (conversor D/A), pois, circuitos de disparos totalmente analógicos são tipicamente mais sensíveis a variação de temperatura, bem como a ajustes de ponto de operação entre outros.

A ponte de tiristores utilizada é do tipo hexafásica, sendo alimentada via um autotransformador pela tensão trifásica da rede de CA de 220 V entre fase. A saída da ponte de tiristores é aplicada diretamente na bobina de campo do gerador.